A model-based safety analysis approach
for high-integrity socio-technical
component-based systems

Master Thesis for the Degree of Master of Science in Computer Science

Master student:
Edin Sefer (esr13004@student.mdh.se)

Supervisors:
Barbara Gallina, Mälardalen University, Sweden (barbara.gallina@mdh.se)
Henry Muccini, University of L’Aquila, Italy (henry.muccini@univaq.it)

Examiner:
Kristina Lundqvist, Mälardalen University, Sweden (kristina.lundqvist@mdh.se)

November 2015
Abstract

Designing high-integrity socio-technical systems requires a thorough understanding of all safety risks of such systems. For many years, safety risk assessment has been conducted separately for hardware, software, human, organizational and other entities in socio-technical systems. Safety risk assessment that does not consider all factors at the same time cannot adequately capture the wide variety of safety risk scenarios that need to be considered.

This thesis proposes a model-based analysis approach that allows interpretation of humans and organizations in terms of components and their behavior in terms of failure logic. The proposal is built on top of the tool-supported model-based failure logic analysis technique called CHESS-FLA. CHESS-FLA supports the analysis of the component-based system architectures to understand what can go wrong at a system level, by applying failure logic rules at a component level. CHESS-FLA addresses only hardware and software components and as such it is inadequate for the analysis of socio-technical systems.

This thesis proposes an extension of CHESS-FLA based on the preexisting classification (developed within SERA), of failures of socio entities. This extension combines CHESS-FLA and SERA - classification and delivers an approach named Concerto-FLA. Concerto-FLA is fully integrated into the CONCERTO framework allowing an automated analysis to be performed on architectures that contain human, organizational and technical entities present in socio-technical systems.

The use of the approach is demonstrated on a case study extracted from the petroleum domain. The effectiveness of the delivered tool is briefly evaluated based on the results from the case study.
Acknowledgments

First of all, I would like to express my deep gratitude to my supervisor at Mälardalen University, Barbara Gallina, for her precious suggestions, guidance, patience and limitless support during the entire thesis period.

Also, special thanks to my supervisor at University of L’Aquila, Henry Muccini, for his constructive comments and generous support throughout my thesis work.

I would like to thank Atle Refsdal for his valuable expertise from petroleum domain offering consultations, constructive discussions and industrial examples for this thesis.

Furthermore, sincere thanks to GSEEM project, especially coordinator Henry Muccini, for allowing me to study at two respectable universities such as University of L’Aquila and University of Mälardalen, offering me to enjoy student life at two wonderful cities, L’Aquila and Västerås.

Also, I would like to give a special thanks to European Commission and EUROWEB project for their generous financial support during my master studies.

Special thanks to my wife Elsada, who always stood by me and constantly strengthened me to believe in dreams and to live dreams. Without her, I would never reach so far.

Last but not least, special thanks to my family for given support and hope during this challenging experience.
Contents

1. Introduction ................................................................................................................................. 7
   1.1. Motivation ............................................................................................................................... 7
   1.2. Context .................................................................................................................................... 7
   1.3. Contribution ............................................................................................................................ 7
   1.4. Document organization ........................................................................................................... 8
2. Background and related work ...................................................................................................... 9
   2.1. Dependability and safety ........................................................................................................ 9
      2.1.1. Preliminary concepts ........................................................................................................ 9
      2.1.2. Dependability and its threats ........................................................................................... 9
      2.1.3. Dependability means ...................................................................................................... 11
   2.2. Component-based reasoning .................................................................................................. 11
      2.2.1. Basic concepts ............................................................................................................... 11
      2.2.2. Component model ......................................................................................................... 12
   2.3. Model-driven architecture ...................................................................................................... 13
      2.3.1. Model transformation types ............................................................................................ 14
   2.4. CONCERTO framework ......................................................................................................... 15
      2.4.1. Separation of concerns in CONCERTO .......................................................................... 15
      2.4.2. CHESS-FLA .................................................................................................................... 16
      2.4.3. Acceleo model transformation engine .............................................................................. 18
      2.4.4. Acceleo MTL ..................................................................................................................... 18
   2.5. State-of-the-art of tool-supported techniques for modelling and analysis of failure behavior of component based architectures ................................................................................. 21
   2.6. Socio-technical systems ......................................................................................................... 21
      2.6.1. State-of-the-art of the risk assessment techniques ............................................................. 22
      2.6.2. Systematic Error and Risk Analysis ................................................................................. 26
3. Petroleum domain-related system ............................................................................................. 30
4. Scientific method ....................................................................................................................... 31
5. Problem formulation and analysis ............................................................................................... 32
   5.1. Problem formulation .............................................................................................................. 32
   5.2. Problem analysis .................................................................................................................... 33
6. Solution ......................................................................................................................................... 35
6.1. Modelling support for failure behavior within socio-technical systems .................................. 35
6.1.1. What should be modeled? ........................................................................................................ 35
6.1.2. Which socio entities should be modeled? ................................................................................. 35
6.1.3. Which existing approach should be used as a basis for modelling of failure behavior of socio entities? ................................................................................................................................. 36
6.1.4. Which technique for analysis of failure behavior should be used? ........................................ 39
6.2. Proposed solution – Concerto-FLA ............................................................................................ 40
6.3. Failure types that characterize the failure behavior of socio entities ........................................... 44
6.3.1. Organizational preconditions in terms of CHESS-FLA FPTC-failure types .......... 44
6.3.2. Human preconditions in terms of CHESS-FLA FPTC-failure types ......................... 46
6.3.3. Human failures in terms of CHESS-FLA FPTC-failure types ............................................ 47
6.4. Automated fine grained analysis of failure behavior of socio-technical entities .................. 54
6.4.1. Proposal for extensions in CONCERTO-ML ................................................................. 54
6.5. Integration of the proposed solution into the CONCERTO tool-set ..................................... 55
6.5.1. Updates of M2T transformation .............................................................................................. 55
6.5.2. Update of backpropagation of analysis results ................................................................. 58
7. Case study ...................................................................................................................................... 60
7.1. Modelling of the case study ......................................................................................................... 60
7.1.1. Modelling internal structure organizational composite block .............................................. 61
7.1.2. Modelling internal structure human composite ................................................................. 62
7.2. Modelling of failure behavior and automated failure behavior analysis ............................... 63
7.3. Discussion .................................................................................................................................. 67
8. Conclusion and future work ............................................................................................................ 69
8.1. Summary ...................................................................................................................................... 69
8.2. Limitations and future work ....................................................................................................... 70
References ........................................................................................................................................... 71
Appendix A: SERA - Linking active failures with pre-conditions ................................................. 75
### Acronyms and Abbreviations

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AEB</td>
<td>Accident Evolution and Barrier Function</td>
</tr>
<tr>
<td>AADL</td>
<td>Architecture Analysis and Design Language</td>
</tr>
<tr>
<td>C2S</td>
<td>Command, Control and Supervision</td>
</tr>
<tr>
<td>CIM</td>
<td>Computationally Independent Model</td>
</tr>
<tr>
<td>CONCERTO</td>
<td>Guaranteed Component Assembly with Round Trip Analysis for Energy Efficient High-integrity Multi-core Systems</td>
</tr>
<tr>
<td>Concerto-FLA</td>
<td>CONCERTO - Failure Logic Analysis</td>
</tr>
<tr>
<td>CHESS</td>
<td>Composition with guarantees for High-integrity Embedded Software components aSsembly</td>
</tr>
<tr>
<td>CHESS-FLA</td>
<td>CHESS - Failure Logic Analysis</td>
</tr>
<tr>
<td>FRAM</td>
<td>Functional Resonance Accident Model</td>
</tr>
<tr>
<td>HERA</td>
<td>Human Error in Air Traffic Management</td>
</tr>
<tr>
<td>HFACS</td>
<td>Human Factors Analysis and Classification System</td>
</tr>
<tr>
<td>HiP-HOPS</td>
<td>Hierarchical Performed Hazard Origin and Propagation Studies</td>
</tr>
<tr>
<td>MDA</td>
<td>Model-Driven Architecture</td>
</tr>
<tr>
<td>MDE</td>
<td>Model-Driven Engineering</td>
</tr>
<tr>
<td>PIM</td>
<td>Platform Independent Model</td>
</tr>
<tr>
<td>PSM</td>
<td>Platform Specific Model</td>
</tr>
<tr>
<td>M2M</td>
<td>Model-to-model transformation</td>
</tr>
<tr>
<td>M2T</td>
<td>Model-to-text transformation</td>
</tr>
<tr>
<td>MTO</td>
<td>Man-Technology-Organization</td>
</tr>
<tr>
<td>RCA</td>
<td>Root Cause Analysis</td>
</tr>
<tr>
<td>SERA</td>
<td>Systematic Error and Risk Analysis</td>
</tr>
<tr>
<td>STAMP</td>
<td>System-Theoretic Accident Model and Processes</td>
</tr>
<tr>
<td>STS</td>
<td>Socio-technical systems</td>
</tr>
<tr>
<td>T2M</td>
<td>Text-to-model transformation</td>
</tr>
</tbody>
</table>
1. Introduction

This chapter represents the thesis introduction which is organized as follows: Section 1.1 describes the motivation for the thesis. Section 1.2 presents the context in which the thesis is defined. Section 1.3 presents the contribution of the thesis and Section 1.4 describes briefly the structure of the thesis.

1.1. Motivation

Work procedures on the offshore petroleum installations (called rigs) often involve safety-critical operations whose failure could cause accidents with serious consequences in terms of hazards to human life, property and environment. Such systems, which involve safety-critical operations, are called high-integrity systems [1] which include socio-technical systems (STS) as well. To keep these systems safe, it is necessary to identify what may go wrong and the way in which this can happen. This represents the essential part of the safety risk assessment and accident prevention. Accident analysis and investigation results showed that in almost 80% [2] of the cases a root cause for accidents in high-hazard industries is the human factor. However, human error occurs in conditions created by organization and environment. Focusing only on one of the contributing factors will not adequately capture all possible scenarios. Therefore, it is necessary to understand interrelatedness of all playing factors and identify the risks in STS.

1.2. Context

This thesis is defined in the context of the CONCERTO (Guaranteed Component Assembly with Round Trip Analysis for Energy Efficient High-integrity Multi-core Systems) [3] project, that strives to provide a multi-domain architectural framework that allows engineers to address dependability concerns, with special focus on safety. The attention of this thesis is limited to the modelling and analysis of safety-critical socio-technical systems. More specifically, within the CONCERTO, the failure behavior related to humans, organizations and technology shall be modelled and the eventual propagation of failures shall be analyzed. One of the supported domains that define the context of this thesis is the petroleum domain.

1.3. Contribution

The contribution of the thesis is related to the modelling and analysis of the failure behavior of socio-technical systems. The current state-of-the-art shows a lack of the tools and techniques for the joint modeling and analysis of failure behavior of socio-technical systems. This thesis contributed to the CONCERTO toolset by developing a method named Concerto-FLA. The novelty of the contribution is that Concerto-FLA permits architects to interpret humans and organizations in terms of composite components, and their behavior in terms of failure logic. Expression in terms of failure logic allows using the existing failure behavior
analysis algorithm in CONCERTO tool-set. The Concerto-FLA represents an evolution of the CHESS-FLA towards support for the analysis of socio-technical systems. This evolution is based on the existing failure classification for socio entities found in SERA approach, combined with the already used failures for technical entities in the CHESS-FLA. Built on top of the Model-Driven Engineering (MDE) premises, the Concerto-FLA uses model-to-text transformations to extract information about failure behavior from models and transform it into the form suitable for failure logic analysis. Moreover, as a result of an early work on this thesis, a paper [5] has stemmed from this thesis and it was presented at 2nd International Workshop on Risk Assessment and Risk-driven Testing (RISK 2014).

1.4. Document organization

Chapter 2 presents the background that is needed for a better understanding of the problem. Concepts presented in this chapter are also used to build the proposed solution in Chapter 5.

Chapter 3 presents the petroleum domain related example that serves to demonstrate the use of the proposed Concerto-FLA.

Chapter 4 presents the scientific methods used in this thesis to find a solution to the addressed problem.

Chapter 5 describes the problem formulation and analysis. The main problem is described in details, divided into smaller subproblems, where each of them is covering a certain aspect of the main problem.

Chapter 6 describes the steps on how existing approaches might be used to solve the identified subproblems. Available approaches are compared and their limitations and advantages are presented with respect to their applicability to the presented sub-problems. A solution to the identified problem is proposed. Chapter 6 also shows how the selected approach is interpreted in terms of the CONCERTO framework and how it is integrated in the CONCERTO tools-set.

Chapter 7 shows the applicability of the implemented solution on a case-study from the petroleum domain, presented in Chapter 3.

Chapter 8 contains the summary and the possible future work related to the results of this thesis.
2. Background and related work

This chapter provides background that helps in better understanding the content of this thesis. It starts with concepts and terms related to the system dependability and safety in Section 2.1. Section 2.2 describes system decomposition in a light of component-based reasoning in software engineering. Section 2.3 describes model-driven approach in architecting systems using model transformations. Section 2.4 introduces reader to the CONCERTO framework and related techniques. Section 2.5 presents the state-of-the-art of approaches that delivered accident models. Section 2.6 is devoted to socio-technical systems and model-based approaches for risk assessment of such systems.

2.1. Dependability and safety

This section is divided into three subsections: Subsection 2.1.1. recalls preliminary definitions needed to understand dependability; Subsection 2.1.2. recalls the definitions of dependability and describes the threats that can affect dependability, while Subsection 2.1.3 presents the general classification of dependability means. This Section is mainly based on [6].

2.1.1. Preliminary concepts

Prior to recalling the concept of dependability, it is necessary to recall the concept of correct service. This subsection recalls definitions of preliminary concepts such as system, its environment, and the service that this system offers.

A system is defined as any entity that interacts with its environment.

Environment of the system is again a system that can be hardware, software, humans or a physical world.

The system acts as a provider that delivers the services to its users or environment. Therefore, a system behavior, seen from a perspective of its users, represents the delivered service.

If a system performs well and if it implements system functions as described in the system functional specification, then it is delivering the correct service.

2.1.2. Dependability and its threats

To rely on a system and to become dependable on a system’s functionality a user expects to receive a service that has certain characteristics. In other words, a user has to trust that the system is going to deliver the correct service.

Dependability is defined as the “ability to deliver the service that can justifiably be trusted”. [6] Another definition of dependability is the “ability to avoid service failures that are more frequent and more severe than is acceptable”. [6]
From a user’s point of view, in order to be dependable, a system is expected to have the following attributes: availability, reliability, safety, integrity, and maintainability. Depending on the system nature and specification, not all dependability attributes are relevant for all systems. In order to make systems dependable, system architects must know the threats to the dependability and in which way to cope with these threats.

Faults, errors and failures represent threats to dependability. They are making a linear chain in which activated faults are causing the errors (see Figure 1). Errors can be propagated to the failures and a failure can cause a fault in another component or system.

According to [6], a fault is adjudged or hypothesized cause of an error.

![Figure 1 - The fundamental chain of dependability and security threats [6]](image)

Faults can be in active or in dormant state, where only activated faults can cause an error and the remaining faults stay in dormant state. Faults can also be internal or external of the system. For an external fault to happen, there must be an internal fault that enables external faults to cause an error.

By activating a fault, a transition from a valid system state to an erroneous state is performed.

An error is a state of the system that may lead to failure to deliver a correct service.

By performing corrective actions, a system can transit from an erroneous state back to the valid state, making the service failure never happen. All these transitions are performed inside the system, transiting between the internal states.

A failure is an event that makes the system transit from an internal erroneous state to an erroneous state that can be perceived from the user.

The ways in which a service can deviate from the correct service represent failure modes of a service.

Failure modes are an interesting field of research for engineers since they do not only describe a possible set of service failures, but also provide a direction for taking necessary actions in making a system more dependable. Failure modes can be seen from four main viewpoints relevant for one safety engineer: failure domain, detectability of failures, consistency of failures and consequences on the environment.

Failures that can rise from the domain viewpoint can be put in three categories:

- Content failure - the delivered content deviates from the specified content to deliver a system function. Content can be expressed by numerical and alphabetical sets.
Timing failure - arrival time or duration of the delivered information deviates from the specification to deliver the correct service. Timing failure can manifest as early or late, depending whether the information is delivered too early or too late.

Content and timing - combined together, these two failures can be classified as halt failures (occurs when a system is halted and has constant external states including no service at all) and erratic failures (occurs when service is not halted, but provides erratic information).

2.1.3. Dependability means

In order to make a system that can be justifiably trusted to deliver a correct service, various tools, methods and solutions are developed. They help architects at coping with dependability threats. These tools, methods and solutions are called dependability means and they are classified as follows:

- fault prevention - focuses on the analysis and adaptation of the development process to reduce possible introduction of faults;
- fault tolerance – focuses on the implementation of techniques to detect faults and recover a system by making it transit from an erroneous state to an error-free state;
- fault removal - focuses on performing techniques to detect and remove faults in the development and use phase;
- fault forecasting – focuses on providing techniques to evaluate system behavior to estimate present and future faults, and to determine their consequences.

This thesis aims to contribute in delivering a fault forecasting technique that helps engineers in identifying the things that may go wrong and the way in which they can go wrong within the socio-technical systems.

2.2. Component-based reasoning

In this section, concepts of the component-based engineering such as component, interface and connector are recalled. The section is organized as follows: basic concepts are presented in Subsection 2.2.1; Subsection 2.2.2 describes the component models and its elements.

2.2.1. Basic concepts

Due to the increasing complexity of the modern systems, system decomposition is used to break a system into smaller building blocks called components. The definition of a system, which was recalled in Subsection 2.1.1, implies that a system can be any entity that interacts with its environment, and therefore a system can be composed of components of different types. Computer systems are typically composed of software and hardware components. A software component can be defined as follows:

“A software component is a unit of composition with contractually specified interfaces and explicit context dependencies only. A software component can be deployed independently and is subject to composition by third party.” [7]
As the above definition states, a component is a unit of composition. Therefore it has to be well-specified and have predictable behavior to enable composition with the other components and integrate it into a system.

"Component composition is the process of integrating components with each other, and with specially written 'glue code' to create a system or another component. “ [8]

The component composition is achieved by using interfaces and connectors between those interfaces. A connector defines the information flow between components. An interface is defined as follows:

"Interface can be defined as a specification of component's access point." [9]

Interface only specifies the possible ways of the component usage by exposing the operations that can be used, but offering no implementation of these operations. This allows the change of the component implementation without changing the interface and affecting the environment of the component.

The component can have two general types of interfaces:

- **Exported interface** - when the component provides a service to the environment;
- **Imported interface** - when the component requires a service from the environment to implement a service.

Unfortunately, interfaces are exposing only functional properties of the component through signatures, while extra-functional properties of the component remain hidden and need to be examined either via credentials or by the analysis of documentation.

**2.2.2. Component model**

Component models have risen from the need to standardize the way of specifying components and its characteristics. One of the general definitions of component model is:

"A **Component Model** defines standards for (i) properties that individual components must satisfy and (ii) methods, and possibly mechanisms, for composing components." [10]

However, many domains have specific requirements to achieve system properties such as real-time and dependability. The general purpose component models cannot be applied on such domains, which led to the existence of many domain specific component models. Domain specific component models incorporate component-based principles with the domain requirements.
2.3. Model-driven architecture

Model Driven Architecture (MDA) is a set of tools and standards that guide system architects in developing robust architectures by separating business logic from the details of the implementation technology. MDA is a subset of a broader concept called Model Driven Engineering (MDE) [11]. Even though CONCERTO is based on the MDE premises, it also implements the MDA principles and structuring of the system models. Therefore, the attention of this thesis is limited on the MDA, also because the main contribution of the thesis is more visible if it is placed into the context of the models and transformations in the MDA.

*MDA* “unifies every step of the development of an application or integrated suite from its start as a Platform-Independent Model (PIM) of the application business functionality and behavior, through one or more Platform-Specific Models (PSMs), to generated code and a deployable application.” [12]

MDA recommends three abstract models to be created: CIM (Computation Independent Model), PIM (Platform Independent Model), and PSM (Platform Specific Model).

*CIM* is a model that is computationally independent and includes domain specific abstractions used in a system. It does not discover the details about system structure, or how the system functionality is realized through model or artifacts. [13]

“*PIM* is a model with a high level of abstraction that is independent of any implementation technology.” [14]

PIM model has a certain degree of platform independence that fits many similar platforms. It shows computational components and their interactions without implementation related details, allowing generating various PIMs. PIM allows managing the application logic across all platforms, since implementation details are separated from the application logic.

“A *PSM* is any model that is more technology specific than a related PIM.” [13]

PSM is the result of the transformation applied on PIM model. Model transformation is a technique that has many practical applications in automating tasks in software development process and it is defined as follows:

*Model transformation means performing a set of rules defined in transformation specification to convert source model or artifact into a target model or artifact.* [13, 14]

Transformation specification defines rules for mapping source representation into target representation. In turn, transformation rule defines how source building elements should be transformed into target elements.

Therefore, PIM to PSM transformation contains additional information that enriches the PIM model with platform specific data and produces a PSM model. Each PSM corresponds to one
specific platform, and if PSM is detailed enough, it allows code generation from model. The contribution of this thesis is placed into the context of PIM, taking it as a source model for the safety analysis and risk assessment.

### 2.3.1. Model transformation types

In model-driven approach, all models, which include a transformation as well, conform to the appropriate metamodels, thus a transformation, source, and a target models conform to their metamodels. A transformation model is defined as a set of transformation rules. The transformation process begins when the transformation rules, expressed using the transformation language, are interpreted by the transformation engine. Then, the transformation engine applies these rules on elements in the source model in order to produce a target model.

In general, it is possible to distinguish the following transformation types and associated technologies [15]:

- **Model-to-model (M2M)**

In the model-to-model transformation type, the transformation engine takes a model as an input, and produces a model as an output. M2M transformations are used to generate different system views for desired viewpoints by hiding information from the source model or enriching the source model with new information. The source code written in programming language can be seen as a model, where metamodel is defined by programming language rules. A good example of M2M transformation can be found in the MDA software design approach, in which PSM is generated from PIM by enriching PIM with platform specific information. Many M2M transformation languages and engines exist today and here is a short list of some of them: EMF, ATL, QVT, Smart QVT, ETL, etc.

- **Model-to-text (M2T)**

In model-to-text transformation type, the transformation engine takes a model as an input, but, instead of a model, as an output produces a text that can be code, text, document or any other form that is not a model. It does not require any metamodel for the output file. M2T is usually used to speed-up software development process, more specifically, to generate code or text, i.e. documentation, from platform specific models. It is necessary to specify transformation for each programming language, or for each desired output text format. Some of the M2T transformation engines existing today are: xText, Acceleo, JET, Xpand, EMF text, Kermeta, MOF script, etc. This thesis uses Acceleo M2T transformation engine to extract extra-functional properties from PIM models on different levels of abstraction. Acceleo is described in more details in Subsections 2.4.3 and 2.4.4.

- **Text-to-model (T2M)**

In text-to-model transformation type, the transformation engine takes a text as an input, and produces a model as an output. T2M is usually used to extract models and metamodels from a
textual representation in order to perform a more complex operation on extracted models. Some of the T2M transformation engines are: EMF text, xText.

2.4. CONCERTO framework

This section introduces reader to the CONCERTO framework and related techniques. The section is organized as follows: Subsection 2.4.1 presents and briefly explains separation of concerns in CONCERTO. Subsection 2.4.2 presents CHESS-FLA as a tool within CONCERTO framework. Subsection 2.4.3 presents Acceleo model transformation engine used in the implementation of CHESS-FLA. Subsection 2.4.4 presents the syntax and use Acceleo modelling language.

As it is stated in the Chapter 1, this thesis is defined within a framework of the CONCERTO project [3]. The work product of the CONCERTO project is the CONCERTO tool-set that has a goal to enhance the CHESS component model to support multi-domain component models, operation modes, and to broaden support for separation of concerns towards a multi-domain environment. The CHESS tool-set is implemented as a set of Eclipse plug-ins that forms a model-driven engineering infrastructure for congregation of high integrity component-based systems. Following the MDA standard, systems are designed through several model-transformations starting with the creation of PIM using the modelling language CHESS-ML. The PIM model is then enriched with platform specific data which allows automatically performing the transformation and generating the PSM model. Finally, a code can be automatically generated from the PSM models. Additional model transformations and a back-propagation mechanism can be used to analyze the models and to guarantee that models and their implementation satisfy the desired extra functional properties.

2.4.1. Separation of concerns in CONCERTO

Architecting of the system using the CONCERTO design process is organized through several multi-level design views. Each view addresses one or more stakeholders and their concerns. In that way, separation of concerns is achieved, by exposing in each view characteristics interesting for stakeholders. At the moment, a CONCERTO environment has the following top level design views:

- **System views** - allows the system engineer to model the user context, business concepts, use-cases, scenarios, product lines variability, analysis and of course the overall, top level system design.
- **PIM views** - represent a set of platform independent views used for the development of the multi-concern component-based architectures. It consists of the following sub-views:
  - **Component view** is used for the modelling of software components. This view is composed of two sub-views: **Functional and Extra Functional views**. The **Functional view** allows specification of functional properties of the components. Components are connected using client-server ports, which are also used to expose previously defined, provided or required interfaces and
data types. In Extra Functional view a user can specify extrafunctional properties of components such as dependability. Information specified in this view can be used in Analysis view to calculate extrafunctional properties.

- **Instance view** – allows designers to perform specification or refinement at instance level. It focuses on instance model which can be automatically derived using information present in composite diagrams.

- **Deployment view** allows the creation of the mandatory target execution platform. This view is composed by the following sub-views: the Platform Specification View, the Allocation View and the Dependability View. Platform Specification View allows the specification of the hardware platform entities. The Allocation View allows designers to allocate the software component instances to the hardware instances which can be enriched with dependability attributes inside Dependability View.

- **Analysis view** allows definition of behavior of the element instances that will be used in the context of the input for one of the available analysis tools. The analysis view is composed of two sub-views: Dependability view (addresses dependability properties of the system) and RTAnalysis view (addresses real-time system analysis).

- **PSM views** – owns the automatically generated entities related to the target computational model. These entities implement component's extra functional constraints defined in PIM. The entities in the PSM View are read-only and can be navigated in the CONCERTO editor by using a tree-like model explorer.

- **Requirements view** allows modeling and tracing functional and extra-functional, software and hardware requirements and connecting them to the model entities like component implementations.

### 2.4.2. CHESS-FLA

This subsection is partially based on [5], that presents initial results connected to this thesis work. CHESS-FLA[27] is a plugin within the CHESS tool-set that allows safety engineers to perform an early based safety analysis on high-level models (PIM). It allows decoration of component-based architectural models with dependability related information which then serves as a basis for the Failure Logic Analysis (FLA) techniques such as FPTC [28] and FF4FA [29]. The analysis results are then back-propagated into the original model. CHESS-FLA supports the analysis of architectures composed of hardware and software components. This thesis focuses on the FPTC (Failure Propagation Transformation Calculus) analysis technique that allows qualitative dependability analysis of the component-based systems in terms of failure propagation and transformation calculus.

The FPTC analysis calculates system behavior based on the specified behavior of its internal components. Therefore, each component has to be examined in isolation to determine its behavior, nominal as well as faulty. A component can act as a source (due to the activation of internal faults, the component generates a failure in output) and as a sink (detection and correction of the input failure). Furthermore, a component can propagate the failure (failure is propagated from input ports to the output ports) and also transform the failure (the nature of the failure has changed and sent on output port). The FPTC failure behavior of the individual component is specified as a set of logical rules expressed as combinations of input failures...
(occurring on input ports) with the output failures (occurring on output ports). CHESS-FLA has adopted the following syntax to specify the behavior of the component:

\[
\text{behavior} \quad = \quad \text{expression} + \\
\text{expression} \quad = \quad \text{LHS} \rightarrow \text{RHS} \\
\text{LHS} \quad = \quad \text{portname} \cdot \text{bL} | \text{portname} \cdot \text{portname} \cdot \text{bL} (, \text{portname} \cdot \text{bL}) + \\
\text{RHS} \quad = \quad \text{portname} \cdot \text{bR} | \text{portname} \cdot \text{portname} \cdot \text{bR} (, \text{portname} \cdot \text{bR}) + \\
\text{failure} \quad = \quad \text{‘early’} | \text{‘late’} | \text{‘commission’} | \text{‘omission’} | \text{‘valueSubtle’} | \text{‘valueCoarse’} \\
\text{bL} \quad = \quad \text{‘wildcard’} | \text{bR} \\
\text{bR} \quad = \quad \text{‘noFailure’} | \text{failure}
\]

The failure modes recalled in Subsection 2.1.3 are further expanded in CHESS-FLA FPTC and can be categorized as follows [19]:

- Value (content) related failures:
  - valueSubtle – “The output deviates from the expected range of values in an undetectable way (by the user)” [29]
  - valueCoarse – “The output deviates from the expected range of values in a detectable way (by the user)” [29]

- Time related failures:
  - early – an output provided earlier than expected
  - late – an output provided later than expected

- Provision (content and time) related failures:
  - omission – no output is provided
  - commission – an output is provided when not expected.

It is important to highlight that valueCoarse is in general detectable by humans and they can perform corrective actions and stop further failure propagation. An example of a FPTC compliant expression is:

\[\text{R1.noFailure} \rightarrow \text{P1.valueCoarse}\]

The above transformation rule is interpreted as follows: if the component receives on its port R1 a ‘noFailure’ which means a normal behavior, it generates on its output port P1 a ‘valueCoarse’ (i.e. value failure detectable by humans).

Connected components are considered as token-passing network in which a system-level behavior is calculated via fixed-point calculation that calculates the maximal-tokenset on any connection in the network. The FPTC technique combines two traditional risk identification techniques (i.e. Fault Tree Analysis and Failure Modes and Effects Analysis) often recommended by safety standards. CHESS-FLA uses ACELEO engine to perform M2T transformations for extracting information about components, ports, connections and the transformation rules. The coproduct of the transformation is XML file which is used as an input to the analysis. It is important to highlight that current CHESS-FLA does not distinguish component types at XML level. Analysis results are also stored into XML file which is then parsed and used for backpropagation of the analysis results.
2.4.3. Acceleo model transformation engine

Acceleo[16] is a M2T transformation tool, developed by Obeo company, whose primary purpose is code generation. It takes any MOF model and a corresponding metamodel as inputs to generate text. Acceleo model transformation engine has been built on top of the OMG MOF Model-to-text specification [17]. It is a part of the Eclipse IDE standard edition and allows code generation from most model types. Acceleo transformations can be run inside Eclipse IDE, but a great advantage of Acceleo is that it can be run as a standalone application. Transformations can be run in background. Hence the end user does not have to know if the Acceleo is used in Eclipse plugin implementation. It is used as a transformation tool in CHESS and as well in CONCERTO. The Acceleo tool is supported by the Acceleo Model Transformation Language (MTL) which is described in details in Subsection 2.4.4.

2.4.4. Acceleo MTL

Acceleo generator is composed of modules declared in .mtl file. Acceleo transformations are written in MTL language. The .mtl file structure begins with module declaration and specification of the metamodels for input models in form:

```
[module <module_name>('metamodel_URI_1', 'metamodel_URI_2')/].
```

A module can extend and import another module and access its public elements. The Acceleo module contains templates, which are generating code, and queries, used for complex expressions. Coding is intuitive and allows plain text, which will be contained in output files, to be mixed with the Acceleo code and rules. Acceleo code consists of the expressions, that can be defined as blocks or inline expressions. Blocks start with [name_of_block] and ending with [/name_of_block], while inline expressions are defined in one line and look like [expression/].

After module declaration, a template is defined with the root element, delimited by [template...][/template] tags. Acceleo transformations are implemented as a set of rules declared inside a template. Rules are matched against a source model in order to generate the output. Templates are the most important constructs in Acceleo since they are giving instructions to the generator while generating the output. They are defined by the template visibility (similar to the visibility in object oriented programming), template name and parameters. There are two types of expressions that can be used inside a template: static expressions and Acceleo expressions. Static expressions are text that will be generated without transformations, while Acceleo transformations are processed by transformations. Templates can be inherited, but inheritance logic differs from inheritance in object oriented programming. It is important to point out that comment @main placed inside the template is used to define the main template which represents a starting point for Acceleo generator.

Inside a template block it is possible to use file tags to generate files as an output. All expressions inside the file block will be written in a specified file. As parameters a file tag has an expression that defines the name of the output file, boolean value that specifies if the output file with the same name exists and should it be overwritten or the results should be
appended to the end of that file. The last parameter is the encoding used for writing a file. More than one file can be generated, but it is recommended to generate one file per module.

Acceleo templates can contain several control structures to control model elements. These structures are if, for, let. If structure allows to execute expressions if a condition is fulfilled. For structure allows iteration through the collections of elements. It is possible to define a separator to insert a text between each iteration. It is also possible to define prefix and suffix by using "before" and "after" functions respectively as parameters to for block. Let block allows a defining variable that can be used in template.

Queries are used to define complex expressions that can be called anywhere in code. Their result is cached and it is recommended to query and cache large iterations. Java services can be called from Acceleo templates or queries. It is recommended to call them from queries so the result can be cached and the transformation will be faster.

An example of using Acceleo transformations is provided below. Figure 2 presents a Papyrus UML model of a very simple system used to demonstrate the usage of the above mentioned Acceleo concepts. The system is named HeatControlSystem and its purpose is to control air temperature in a room using two components: a temperature sensor (modelled as TempSensor component) and a ventilator (modelled as Ventilator component) used to decrease temperature by injecting colder air into the room. These two components are linked together through a connector where TempSensor component sends temperature data to the Ventilator component.

![Figure 2 - Papyrus UML model of simple system named HeatControlSystem](image)

An Acceleo code that transforms this UML model into a XML formatted file is shown in Figure 3. First four lines in Figure 3 are defining the module name and the template that receives a model. The next two lines (lines 5 and 6) are defining the output file and the header of the xml file. After that, in the first for loop, the template is iterating through all components until it does not find the HeatControlSystem component. Then in second for loop, transformation takes all its subcomponents as Property and prints components and its name and type. Third for loop prints all component ports with its name. The last for loop prints connectors with the source and destination ports. Figure 4 shows the generated output of the Acceleo transformation showed in Figure 3.
```java
[template public generateXMLFromUML(model : Model)]
[main]
[if (model.name.contains('HeatControlSystem'))]
[for (prop : Property | model.eAllContents(Property))]
[Component name="[prop.name/]"] type="[prop.type.name/]">
[Ports]
[for (port : Port | prop.type->filter(port.name/))]
[Port name="[port.name/]"]
[/for]
[/Ports]
[/Component]
[/for]
</Component>
</Components>
</Connector source="[conn.end.role.name->sep('" destination="")/]">
[/for]
</Connectors>
[/if]
[/for]
</Components>
[/file]
[/template]
```

Figure 3 - Acceleo template for transformation UML model to XML

```xml
<?xml version="1.0" encoding="UTF-8"?>
<Components>

  <Component name="HeatControlSystem" composite="true">
    <Component name="TempSensor" type="TempSensor">
      <Ports>
        <Port name="P1" />
      </Ports>
    </Component>
    <Component name="Ventilator" type="Ventilator">
      <Ports>
        <Port name="P1" />
      </Ports>
    </Component>
  </Component>
  <Connectors>
    <Connector source="P1" destination="P1" />
  </Connectors>
</Components>
```

Figure 4 - Model.xml file generated as a result of the Acceleo transformation
2.5. State-of-the-art of tool-supported techniques for modelling and analysis of failure behavior of component based architectures

In this section a state-of-the-art of the tool-supported techniques for modelling and analysis of failure behavior of component based architectures is presented. State of the art on this topic was exhaustively explored in [18] and partially in [19] and [20], but it is repeated and updated here for better understanding of the context of this thesis and the state-of-the-art outside CONCERTO. The attention is limited to the tool-supported model-based techniques and tools.

AltaRica [21] is a high-level modelling language, which models systems as linked components. It allows textual and graphical description of system behavior through automata and state diagrams by modelling all possible system behavior on fault occurrence. Modelling and analysis is supported by the same named tool. AltaRica itself does not follow MDA-like design process, but [22] presented a way of transforming AADL to AltaRica.

AADL (Architecture Analysis and Design Language) [23] is an architectural description language designed to support modelling of systems composed of hardware and software entities using the compositional approach. It allows modelling and analysis of failure behavior using error models represented as a state machine. The error model supports the specification of failure types and it is associated with the design model. It allows derivation of system level failure behavior based on the error models of internal components. Tool support is provided via Osate 2 plugins [24].

Hierarchically Performed Hazard Origin and Propagation Studies (HiP-HOPS) [25] is a dependability technique to specify the failure behavior of the components. Failure behavior is modelled using the Interface Focused FMEA annotations associated to each component. Failure propagation behavior is described in annotations as logical expressions. Failures at output are a result of the combination of incoming and internal failures. Integration of the HiP-HOPS into the EAST-ADL2 presented in [26] uses the model transformation to generate the HiP-HOPS inputs from the EAST-ADL2 error models.

FPTC is another technique for failure behavior analysis that is supported by the CHESS-FLA tool. This technique and the tool are already described in more details in Subsection 2.4.2.

The comparison of the aforementioned techniques with respect to the CONCERTO project is provided in Chapter 5.

2.6. Socio-technical systems

The concept of the Socio-Technical Systems was first introduced by Trist [30] in fifties of the previous century. By studying the coal mining process he discovered that a system performance is not dependent only on applied machines, but also on the way humans use and manipulate with machines. This concept was a subject of many studies performed by Trist’s successors, and today it is a very important field of research. Based on [30], Baxter and Sommerville [31] defined socio-technical systems as “systems that involve a complex
interaction between humans, machines and the environmental aspects of the work system”. The term “socio-technical” refers to the interrelatedness of “social” and “technical” [30] and represents a basis on top of which the socio-technical systems design was built. Relations between the social and technical entities are a part of the processes in the systems and organizations and can be both linear and complex (dynamic). Linear relations are described as the straightforward chain reaction trees where top event in the tree is caused by one or more leafs (basic events). Complex relations comprise the dynamic interactions that can change behavior during the time and cannot be expressed using linear techniques. In order to make these systems dependable, it is necessary to make an estimation of the risks that can threaten the dependability. In other words, identifying what may go wrong and the way in which it can happen is of crucial importance in defining the dependability means recalled in Section 2.1.3.

One of the risks to the dependability of STS is human factor. Many accident investigations related to STS showed that in a very high percentage of accidents human factor was either a root cause for an accident, or it was one of the contributing factors to the accident. The accident analysis is usually performed in a way that human factor is mostly investigated on the executive level in the organizations. This usually led to criminalization of the individuals. However, organizations and society are also built of the humans that are involved in decision making process on each hierarchical level in the organizations and in the government. Some authors have delivered accident investigation models which help in modelling the accident and understanding the circumstances in which the accident occurred. In order to capture all the factors involved, those accident investigation models should encompass all the involved entities, starting from the top level government, regulatory, management down to the operational level and the environment conditions. The accident investigation models could also help in defining safety risks and defining safety barriers against the identified risks. Subsection 2.6.1 describes various approaches related to the safety risk assessment of the socio-technical systems.

2.6.1. State-of-the-art of the risk assessment techniques

The failures in socio-technical systems influence humans, often causing accidents with catastrophic consequences. Many techniques for safety analysis have emerged and are often focused on accident analysis and the creation of accident models. Accident models are interesting from the system design perspective since they can give useful models of nominal and faulty system behavior. Accident models show relations between the causes and the effects and “are used as techniques for risk assessment during system development, and post hoc accident analysis to study the causes of the occurrence of an accident”. [32] It is possible to distinguish traditional approaches, systemic approaches and the approaches using formal methods [32, 33].

Traditional approaches, such as Root Cause Analysis (RCA) [34], Domino model [35], Fault Tree Analysis [36] and Five Whys method [37], model the accident as a consequence of the sequential, linear, time ordered events, which implies that once the root cause in the sequence is removed, the accident recurrence is stopped. Root Cause Analysis consists of four steps, namely, (i) gathering the data, (ii) representing events as sequence diagrams, (iii) analysis and
identification of root causes, and (iv) generation of counter measures. The Domino model is modelling events as a chain reaction in the domino failing effect, where each contributing factor is a consequence of the previous one in the chain, ending with the root cause. Fault Tree Analysis is a top down approach that models events as a logical tree, searching for the minimal cut set of the basic events that contributed to the top event. Five Whys is a top down method that builds causation chain by imposing a question “why” on the top event and on each answer provided in the chain. Since these traditional approaches are able to capture only linear relations between socio-technical entities, they cannot be applied on complex systems with dynamic behavior. [32] Also, these approaches are not distinguishing types of entities in STS and do not deliver any classification of failures for socio-technical entities.

AEB (Accident Evolution and Barrier Function) [38] is another example of a sequential approach that captures faulty linear interactions between human, organizational and technical entities and represents them as barriers in a flow diagram. It is important to highlight that AEB models only error event and not correct ones. AEB provides classification of barriers but lacks the classification that can be directly associated to human and organizational entities. HERA (Human Error in Air Traffic Management) [39], on the other hand, focuses on human error as a primary contributor to the accident. HERA addresses linear relations between entities and strives to identify all human error types, conditions of the task, environment and organizational factors. It provides an exhaustive classification of human and some of organizational errors types. However, the provided classification is highly related to avionics domain and it requires a study on applicability on other domains.

Systemic approaches are based on a system theory and see accident occurrence as a consequence of complex relationships and dependencies between all system components (human, technical, organization, environment, social) that can contribute to the failure.

One of the systemic concepts that considered all aspects of STS has been developed in Sweden and was named MTO (Man, Technology and Organization) [40]. The concept is similar to Human Factors (HF) concept developed in the USA and it suggested the holistic system view on system safety. MTO suggests that during safety risk assessment Man, Technology and Organization should not be analyzed separately, but instead they should be seen as one whole and analyzed together.

Several MTO-oriented risk assessment techniques emerged until today. One such approach is Reason’s Swiss cheese model [41] which considered that an accident is produced as a combination of triggered latent conditions and active failures committed by individuals or teams. Active failures and latent conditions are seen as holes on slices of the Swiss cheese, where slices of the cheese represent safety barriers and guards. Once the holes (failures) are lined up, failures will pass all barriers on each level and cause an accident. An approach based on the Reason’s Swiss cheese model is the Human Factors Analysis and Classification System (HFACS) [42] that provides taxonomy of human failures and preconditions on the supervision and organizational levels. HFACS represents a framework for systematical examination of the accidents and their causes. According to Reason, accidents are caused not only by human error but also by propagated latent failures in the upper layers in the
organizational hierarchy. HFACS describes four levels of failure: 1) Unsafe Acts, 2) Preconditions for unsafe acts, 3) Unsafe supervision, and 4) Organizational influences. However, HFACS models linear relation between human and organizational entities and provides fine classification of failures for both human and organizational failures.

Rasmussen’s socio-technical framework [43] adopts a concept of a system safety as a control structure problem based on the control loops through the organizational, management and operational structures. This framework has two parts: a hierarchical model, that describes the structure of the system, and a dynamic model. Hierarchical model (see Figure 5) represents a static view on the system which contains many control loops. Control flows from top layers (management) to bottom layers (executive layers), while at the same time the feedback flows in the opposite direction from bottom layers towards the upper layers. The dynamic model defines the boundaries of safe operations: economic boundary, unacceptable workload, and safety regulations boundary. Workload pressure and financial cost are strengthening the operational space towards safety regulations boundaries, which causes that over a time period, people cross the safety regulations boundary. System can still operate without accident until it does not reach functionally unacceptable behavior. Rasmussen’s socio-technical framework is able to address dynamic and complex relations between all entities in STS, but lacks an appropriate classification of failures of socio-technical entities.

Similarly to the Rasmussen’s approach, STAMP (System-Theoretic Accident Model and Processes) [44] uses control loops to discover violated constraints on each control level. STAMP approach models accident causes not just as independent component failures, but also as external influences and dysfunctional interconnections between the internal components. “Safety can be viewed as a control problem, and safety is managed by a control structure embedded in adaptive socio-technical system.” [44] The first step in accident analysis using STAMP is the development of a hierarchical control structure. The hierarchical control structure is then analyzed to determine the constraints and causes of flawed control at each level. STAMP threatens humans as any other entity in the system and applies the unified approach to all identified functions. The same approach can be used also for risk assessment.
during the system development phase. STAMP successfully addresses complex dynamic relations between entities in STS by focusing on interactions among components. However, the provided classification of failures is capturing failures at control loop level (Figure 6) and can be rather associated to organizational level than to human behavior.

Several techniques based on the formal methods are available as well, but they are focused on the design and verification of systems, and an extension towards socio-technical aspects is needed.

Systemic cognitive systems engineering approaches such as the Functional Resonance Accident Model (FRAM) [45] model systems with human and technology interaction in the context of the workplace and its environment. FRAM addresses complex and dynamic relations in STS using the concept of functional variability. FRAM describes how the coupling and dependencies between system functions can contribute to an accident occurrence through unhandled propagation of human, technological and organizational performance variability. The FRAM risk assessment starts with an identification and characterization of the essential system functions. The characterization can be made based on the six connectors of the hexagonal functional representation (Figure 7): Input, Output, Resources (what is needed by the function to process the input), Controls (serves to supervise or restrict the function), Preconditions (system condition), Time (time constraints such as start and finish time or duration). System functions are interconnected using input and output ports, which in the end form a structure which allows tracking of the failure propagation through functions. However, FRAM models might need more than one connection between functions to express one event which is seriously reducing readability and understanding. The next steps are characterization of the potential variability, definition of functional resonance based of dependencies among functions, and identification of barriers for variability.
Various ways in which an output can differ from the expected values (resonance) can be expressed using failure modes. Also, FRAM considers the common performance conditions (Figure 8) that lead to performance variability of human, technical and organizational functions and defines the likelihood for the variability. However, the common performance conditions might affect functions of all MTO categories and cannot be used to classify the failures of any of the MTO categories.

<table>
<thead>
<tr>
<th>Common performance conditions</th>
<th>Functions affected</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>M</td>
</tr>
<tr>
<td>Availability of resources</td>
<td>X</td>
</tr>
<tr>
<td>Training and experience (competence)</td>
<td>X</td>
</tr>
<tr>
<td>Quality of communication</td>
<td>X</td>
</tr>
<tr>
<td>HMI and operational support</td>
<td>X</td>
</tr>
<tr>
<td>Access to procedures and plans</td>
<td>X</td>
</tr>
<tr>
<td>Conditions of work</td>
<td>X</td>
</tr>
<tr>
<td>Number of goals and conflict resolution</td>
<td>X</td>
</tr>
<tr>
<td>Available time and time pressure</td>
<td>X</td>
</tr>
<tr>
<td>Circadian rhythm and stress</td>
<td>X</td>
</tr>
<tr>
<td>Team collaboration quality</td>
<td>X</td>
</tr>
<tr>
<td>Quality and support of the organisation</td>
<td></td>
</tr>
</tbody>
</table>

Figure 8 - FRAM - Match between MTO categories and common performance conditions [45]

2.6.2. Systematic Error and Risk Analysis

Systematic Error and Risk Analysis (SERA) [46] allows systematic investigation of accidents with a focus on human factors and related preconditions to accidents. It combines Reason’s latent failures model with multi-layered Perceptual Control Loops (PCL) (see Figure 9) and delivers a process for identifying active failures and preconditions that lead to these failures. Active failures are the unsafe acts that can be directly connected to accident. However, SERA does not focus on human error and blame of individuals, but rather on the organizational influences that support human errors. Moreover, SERA addressed the interrelatedness of organizational faults and human faults in terms of direct relation between them (see Appendix A). SERA also considered the impact of human internal preconditions for faults such as physical and psychical states. Also, SERA is based on a mature approach called HFACS.
whose application was studied by many researchers from many different domains ([48], [49], [50], [51]) including the petroleum domain [52]. Initially, SERA was designed to help accident investigators in performing the HFACS analysis, but it has been developed as a standalone analysis approach with its own, even broaden classification of failures.

PCL at human level (see Figure 9) describes the human decision processes as a reaction to feedback received as a perception of real world and human model of the world. SERA establishes a PCL on each of the four levels of the Reason’s latent failures model interpreted as:

- **Active failures** – represented as twelve points of a breakdown in human information processing system;
- **Preconditions for unsafe acts** – preconditions that are directly connected with active failures, defined as condition of the personnel, condition of the task and working conditions;
- **Command, Control and Supervision (C2S)** – related to the management activities for forming the goals and providing feedback;
- **Organizational influences** – represented as organizational factors that create a space for flaws on lower levels in the organizational hierarchy.

Using PCL loops at each defined level, SERA is able to capture dynamic relations between entities in STS. It also provides an exhaustive classification of human and organizational failures at each level and provides a detailed mapping of relations between those failures. SERA also considered the contribution of the state of the human to the accident occurrence. A classification of human and organizational failures is represented as active failures and preconditions that lead to these failures.

Twelve categories of active failures that humans can produce are:

1. **Intent failure** - The intention failure is a result of the action that was performed to achieve the goal that is either not consistent with the rules and regulations (violation) or the goal that is consistent but not appropriate to the human that performs a task;
2. **Attention failure** - Attention failure represents a failure in human internal functionality that manifests as a failure to attend all the relevant information that was present;

3. **Sensory failure** - Sensory failure occurs in the case in which a human does not have the physical capabilities such as hearing, eyesight and other sensory capabilities to perform the task;

4. **Knowledge (perception) failure** - This failure occurs in case when a human does not have knowledge or skills to properly interpret the situation;

5. **Perception failure** - Perception failure occurs when all relevant information was attended, but the content was ambiguous. It can occur due to the human processing nature that can filter the available information;

6. **Communication/Information failure** - Communication failure is the failure in the communication channel between humans or between a human and a machine. Due to a communication channel failure a human could not receive the relevant information, or the wrong or incorrect information could be received;

7. **Time management failure** - Time management failure is “a failure to use appropriate and effective time management strategies, including: incorrect or inappropriate prioritization of attention, failure to delegate, postpone, shed tasks, failure to simplify the task, failure to take control of the timeline of the activity, or a failure to pre-plan or bring tasks forward.” [46];

8. **Knowledge (Decision) failure** - occurs when a human does not have a knowledge or skills to make an appropriate decision required to successfully complete the task;

9. **Ability to respond failure** - A failure in the ability to respond relates to the lack of human physical abilities (e.g. strength, vocal effort) to provide a response required to achieve the task goal;

10. **Action selection failure** - This is “a failure in the decision process due to the shortcomings in action selection, rather than a misunderstanding or misperception of the situation. This includes an inappropriate ‘no action’. ” [46];

11. **Slips, Lapses and Mode errors** - This failure occurs when a response is not performed according to the intention. Slips usually occur when an action slightly differs from the routine. Mode errors are related to actions that are inappropriate for current working mode, but correct for another mode;

12. **Feedback failure** - Feedback failure occurs when a human does not receive any feedback information that could be used for correcting actions. This is a failure in error correction since there is nothing that will acknowledge human that the goal is achieved. It includes a failure at the individual (monitoring, checking) and team (crosschecking, supervision, backing up) level.

The preconditions for unsafe acts describe the condition of:

- **Personnel** – This condition is further broken down and represented as seven states of the individual, working both individually and as a team or a group:
  1. **Physiological** - physiological state of the individual such as drowsiness, medical illness, intoxication, etc.;
2. Psychological - psychological states, attitudes, traits, and processing biases that can affect perception, action selection, or forming an action;
3. Social – factors that influence the interaction among groups and teams;
4. Physical capability – factors related to the physical abilities to sense and perform an action;
5. Personnel readiness – the state of the individual in the sense of a physiological, psychological, physical and mental readiness to perform a task;
6. Training and selection – refers to the skills and knowledge required to do the job;
7. Qualification and authorization – refers to the legal pre-requisites to perform a task such as proper qualification and authorization.

- Task – This condition is further decomposed on two factors:
  1. Time pressure - Time pressure occurs when the pace of the task is extreme or when humans, paced by the task, have little scope to actively manage the task timeline;
  2. Objectives - Objectives failure refers to unclear, inappropriate, inconsistent and risky task objectives.

- Working conditions – These conditions are further decomposed on following factors:
  1. Equipment – This factor refers to a condition of tools used to carry out the task;
  2. Workspace – This factor describes the physical arrangement and layout of the workspace;
  3. Environment - This factor describes conditions of the environment in which the activity is performed.

C2S failures are defined in the following terms:

1. Forming intent – Refers to the unclear forming of the objectives of the task, and the assignment of responsibility by managers and supervisors;
2. Communicating intent – The intent was well formed, but the failure occurs due to the unclear communication of intent;
3. Monitoring and Supervision – a failure in the monitoring or supervision activities manifested as a missing, delayed or inadequate error-correcting activity that ensures a successful task or mission completion.

Organizational influences cover the following six factors:

1. Mission - Mission factor refers to the cases when a mission is not clearly defined, approved, or inconsistent with the available resources;
2. Provision of resources - This refers to the management and provision of the resources that include humans, equipment, money and funds, needed to successfully complete the task;
3. Rules and regulations – refers to the set of rules and regulations that set legal constraints on processes in organization. This factor questions whether procedures, mission, operations and goals are set according to the rules and regulations and
whether standards form sufficient safeguards. Rules and regulations may be imposed by an external body;

4. **Organizational processes and practices** - refers to the way of working that includes appropriate procedures (official or formal statements of how a job has to be done), operations (processes established by the management that determine the characteristics or conditions of work) and change management;

5. **Organizational climate** – Organizational climate is the common atmosphere in the organization that shapes the workers’ and managers’ attitudes about safety and makes certain scenarios likely to happen. The organizational climate is affected by the organizational structure, policies and culture (unspoken rules);

6. **Oversight** – Oversight refers to the existence of methods for risk management, self-study, monitoring, and the supervision of the organizational factors that include resources, climate, and processes that identify and correct deficiencies, and ensure a safe and productive work environment.

### 3. Petroleum domain-related system

This chapter introduces the petroleum domain-related system as the domain of interest for this thesis. Offshore petroleum installations represent an example of complex socio-technical systems that are addressing complex interaction between social (human and organizational) and technical entities. In this chapter, a simplified subsystem is presented. This subsystem was initially presented in [5] and in the context of this thesis will be used to illustrate the applicability of the modelling and analysis approach presented in this thesis. This subsystem concerns gas leakage detection and is a part of an overall safety barrier function to prevent ignition of hydrocarbons on a rig. The information about gas leakage is crucial for welding, replacement of gas detectors or other non-routine work on rigs that may increase the risk for accidents. To reduce a risk, all non-routine work needs to get a work permit prior to its execution. In order to reject or release (accept) a work permit, a decision maker needs to take into account a number of safety-related indicators that include the current state of safety barriers such as information about gas leakage. In general, two ways of gas leak detection are used: automatic gas detection and manual gas detection.

Automatic gas detection is an automated system that consists of electronic sensors placed on the strategic points on a rig, usually close to the junctions of pipes or valves. If the sensors register a gas leak, the information is stored into deviations database and becomes available to the decision maker when issuing a work permit.

In contrary to the automatic gas detection that is always monitoring the gas leakage, manual gas detection is done periodically and in teams. Manual gas detection is typically done simply by smell, or by performing techniques such as spraying soap on the location where gas might leak and check for bubbles. Manual gas detection depends on a number of factors such as:
• training and skills of the personnel – the likelihood for a successful gas detection may be reduced if the personnel is not well trained; meaning there will be a lack of skills or knowledge for performing a working procedure,
• working procedures that are followed – lack of procedures or inappropriate procedures defined for the gas leakage detection may significantly reduce the likelihood of a successful manual gas detection,
• time pressure – if there are many places to check and very little time is available, then it is very likely that the manual gas detection will not be completed on time,
• number of persons - the number of persons on the rig will vary over time. If there are few people, there is a smaller chance that a leak will be detected manually, as there will be fewer “noses” on the rig.

These factors are acting together. If there is a lack of training or there are inappropriate procedures and few persons on the rig, then manual gas detection can fail with omission and miss to detect a gas leak. However, if there is only time pressure and a worker is unable to finish the manual gas detection in specified time, then the results of the manual gas detection can come too late.

On a higher level, if both, the manual gas detection and the auto gas detection fail with omission of gas detection, then the total output of the gas detection system will be omission. Otherwise, if either manual or automatic gas detection is late, the total output is late.

### 4. Scientific method

This chapter describes the process used to come up with the solution to the problems addressed in this thesis. The process is based on [53] which is extended and adapted according to the needs of this thesis. The adapted process is displayed in Figure 10.

The method used in this thesis starts with the identification of the main problem. The main problem is analyzed and decomposed into less complex sub-problems in Chapter 5. The sub-problems are then analyzed in Section 5.2 and questions that need to be addressed are imposed.

Based on the formulated questions a selection criterion for the studies is established. The state-of-the-art is explored by doing a search of the existing studies for imposed question. The selection criteria are applied on the search results to filter the results and narrow attention. The given results are then compared and analyzed to find an answer to the given question. If state-of-the-art is not addressing the question, then an extension of the existing approach is proposed.

After finding answers to the questions, a solution to the addressed problems is proposed in a form of a scientific paper. Implemented solution is then evaluated on a case study from the applicable domain. The results of the evaluation are then critically discussed. If the evaluation results are not satisfactory, the question and solution are revised and analyzed considering the
evaluation results. After the satisfactory evaluation, the conclusion is taken from the evaluation and discussion and the report is written.

Figure 10 – The scientific method used in this thesis

5. Problem formulation and analysis

In this chapter the problem to be solved is described in detail and it is analyzed in order to identify the sub-problems and ease its resolution. This chapter is organized as follows: In Section 5.1 the problem to be solved is presented in detail; In Section 5.2 the problem is analyzed and it is decomposed into less complex sub-problems.

5.1. Problem formulation

As it is mentioned in Section 1.1, identifying the things that may go wrong and the way in which they occur is of crucial importance for understanding the safety of the systems. This requires the analysis of the system behavior which leads to the modelling and description of the failure behavior of the system and its internal components. To be able to describe the failure behavior of the high-integrity socio-technical systems, safety engineers must be able to
model all the factors that can contribute to the accident occurrence. Several techniques are available to address these factors with different focus and they are rarely combined into a unified technique. Therefore, an adequate support for the modelling and analysis of failure behavior within socio-technical systems is needed, which is one of the main problems addressed in this thesis. As mentioned in the Section 1.2, this thesis is defined within the context of the CONCERTO project. Thus, a modelling support and an automated failure behavior analysis have to be provided within the context of the thesis. It is worth to mention that failure behavior analysis of technical entities is already supported in CONCERTO using the CHESS-FLA plugin. In the following section the presented problem is analyzed in more details.

5.2. Problem analysis

The main problem addressed in this thesis, presented in Section 5.1 can be decomposed into two sub-problems:

- **Modelling support for failure behavior within socio-technical systems**
  
  When arguing about a modelling support for STS, it is possible to impose the following questions that need to be addressed:

  - **What should be modelled?**
    
    In order to be able to provide a modelling support for the failure behavior within STS, it is necessary to identify all the entities that are present in STS from a high-level perspective.

  - **Which socio entities should be modelled?**
    
    To support the modelling of the failure behavior of socio entities, it is necessary to thoroughly analyze and understand the concepts that are going to be adopted. The socio entities that should be supported for modelling must be identified. It is necessary to define the level of abstraction and granularity of these entities.

  - **Which failure types characterize the failure behavior of socio entities?**
    
    A failure behavior of entity is expressed through the failure types that describe in which way the entity may fail. To be able to perform the analysis, besides the identified socio entities, it is necessary to identify a set of failure types for each socio entity in the system.

  - **Which existing approach to use for modelling of failure behavior of socio entities?**
    
    For an integrated framework that supports the modelling of the safety concerns related to complex interactions between the socio and technical entities, several techniques can be used. These techniques are offering different advantages: some of them focus on tool-support, some approaches focus only on linear relationships between socio-technical entities, some focus on both linear and non-linear relationships, some focus on only human factor. When considering a suitable approach it is necessary to focus on the accident models which can serve to define the faulty behavior of the socio-technical entities. Due to the
large number of available techniques, a state of the art presented in Section 2.6 has to be analyzed. Furthermore, criteria to select the appropriate technique and compare them must be established. Since the modelling support needs to encompass all types of socio-technical entities, one of the criteria could be the comprehensive modelling support. Another criterion is the existence and depth of the classification of faults for socio entities. The classification of faults describes the failure behavior of the analyzed entities. If the selected approach has some limitations with respect to the presented problem, to cover a lack, modifications need to be proposed.

- **Automated fine grained analysis of failure behavior of socio-technical systems**
  When arguing about the automated failure behavior analysis of socio-technical entities the following questions can be imposed:
  
  o **Which technique for analysis of failure behavior can be used?**
    After the modeling of the failure behavior of the socio entities, a failure behavior analysis is needed to discover the faulty relations and the failure propagation throughout the system entities. Among many techniques presented in Section 2.4 the most suitable one with the modelling choice should be selected.
  
  o **Which tool supported technique can be extended to support for the modelling and analysis of failure behavior of socio-technical systems?**
    For a technique to be accepted and used, a tool support must be ensured. It is necessary to select an appropriate tool that can be extended to support the modelling and analysis of the STS. Even though this thesis is defined in the context of the CONCERTO project, for better understanding of the thesis itself and the related tools, it is worth to explore the state of the art outside the CONCERTO.
    The currently existing dependability analysis techniques in CONCERTO should be further investigated to define an implementation strategy. Since the CONCERTO project is built on top of the CHESS project, an investigation of the CHESS project heritage is needed as well. Currently, the CONCERTO project supports only the modelling of the software and hardware entities. When integrating the selected techniques into the tool, two things need to be considered: an extension of the tool itself, and also an integration of the selected modelling and the analysis techniques. Therefore, if needed for the implementation purpose, a modification and extension of the CONCERTO framework should be proposed.
6. Solution

This chapter presents solution methods and a solution that addresses the raised questions regarding the modelling and analysis of the socio-technical systems presented in Chapter 5. In Section 6.1 various modelling approaches for failure behavior analysis of socio-technical systems are compared and the most suitable one is selected. In Section 6.2 a solution for the modelling and analysis of the failure behavior of socio-technical systems is proposed. Section 6.3 describes the interpretation of human and organizational failures in terms of the failure modes as the main contribution of the thesis. Section 6.4 describes the proposed changes for enabling an automated analysis of the failure behavior of the socio-technical entities. Section 6.5 describes the integration of the proposed Concerto-FLA into the CONCERTO tool-set.

6.1. Modelling support for failure behavior within socio-technical systems

This Section directly answers questions regarding the modelling and analysis of socio-technical entities imposed in Chapter 5. It is organized as follows: Subsection 6.1.1 describes what should be modelled, Subsection 6.1.2 describes which socio entities should be modelled, Subsection 6.1.3 describes the selection of the technique used as a basis for the modelling of socio entities, Subsection 6.1.4 describes the selection of the technique for analysis of failure behavior of socio entities.

6.1.1. What should be modeled?

As mentioned in Section 2.6, STS refers to the interrelatedness of socio and technical entities. In order to have a comprehensive approach to the safety risk assessment, modelling support for failure behavior of socio and technical entities is needed. Until now, technical entities are thoroughly explored and many tools are available to model failure behavior of software and hardware entities. However, there is a lack of tools and techniques for the full modelling support for failure behavior that supports a joint analysis of all entities in STS. However, there can be many socio entities in STS that need to be addressed. System architects must be able to identify socio entities as well and model their behavior, nominal as well as the faulty behavior. Subsection 6.1.2 clarifies which socio entities should be modelled.

6.1.2. Which socio entities should be modeled?

Many researchers have provided a different answer to this question. Some of them kept a high-level view on socio entities while others provided a fine grained identification of entities present in STS. Among many approaches presented in Subsection 2.6.1, it is possible to distinguish MTO, Rasmussen’s and Reason’s approaches that served as a basis for many researches of their successors.

The MTO approach provides a high-level view on socio-technical entities. MTO is considered as a systemic approach that encompasses three types of socio-technical entities: Human, Organization and Technology. However, these three entities are representing entities with
very complex and dynamic behavior to be modelled as high-level entities. Therefore, it is necessary to further decompose socio entities and model their interrelatedness.

Rasmussen’s hierarchical model identifies six socio entities ordered in layers, where each layer communicates with the one above and one below in hierarchy. Starting from the top of the hierarchy, those layers are: government, regulators, company, management, staff and work. This approach is rather focused on the organizational entities and their hierarchy, while human behavior is considered as a consequence of the input from the higher levels. This approach does not cover interaction between the socio and technical entities.

Reason’s Swiss cheese model identifies human and organizational entity. However, some extensions of this model, such as SERA and HFACS, identify four layers inside an organizational structure where each layer and its sublayer can be represented as an entity. Compared to Rasmussen’s approach, entities in SERA and HFACS, such as regulators and the government, are presented as organizational subentities, while in Rasmussen’s approach, regulators and government are identified as separate entities that interact with the organization. SERA also recognized that human entities have internal factors and states that can influence action and the human decision making process.

However, entities identified in Rasmussen’s approach are mostly related to the organizational influences, where government and regulatory can be considered as external factors to the socio-technical system. Therefore, this approach addressed organizational and external factors. Swiss cheese extensions, SERA and HFACS are providing too coarse grained identification of socio entities for a high level perspective. Therefore, inspired by the MTO approach, in this thesis humans and organizations are modeled as high-level socio entities. Since a high abstraction level is not expressive enough for modelling failure behavior of complex entities such as humans or organizations, high-level entities should be also decomposed onto smaller subentities, considering the internal functionalities of these entities. Approaches such as SERA and STAMP also identify failure types for human and organizational entities. In order to use those failure types for the modelling of the failure behavior, it is necessary to find a suitable way of modeling and analysis. The next subsection is considering the MTO-oriented approaches that can be used to further decompose the identified high-level socio entities and to model their failure behavior.

6.1.3. Which existing approach should be used as a basis for modelling of failure behavior of socio entities?

This subsection describes the process followed to select an adequate modelling approach for the failure behavior of STS. The MTO-oriented approaches stimulate the comprehensive joint analysis of all entities in STS. Therefore, the focus is on the MTO-oriented approaches that proposed accident models. During the selection, available approaches, presented in Subsections 2.6.1 and 2.6.2, are compared and analyzed based on the criteria defined in the problem analysis in Section 5.2.

- **Modelling support**: This criterion is chosen since it defines the expressiveness of the safety engineers when modelling the failure behavior of socio-technical entities.
Approaches that satisfy this criterion should address both human and organizational factors and be able to capture complex relationships between entities. As mentioned in Section 2.6.1, approaches such as AEB, HERA, Domino model, FTA, RCA, Five whys and Swiss cheese address linear relations between socio-technical entities and as such do not satisfy this selection criterion. Also, Domino model, FTA, RCA, Five whys are addressing the relation between causes of the accident without relation to particular human or organizational factors. Swiss cheese approach is focused on addressing organizational factors, considering human failures as a consequence of the organizational failures. FRAM, on the other hand, addresses human and organizational factors as dynamic functional variations which might not reflect a system internal organization. Besides FRAM, also SERA, Rasmussen’s hierarchical approach and STAMP address both human and organizational factors and are able to capture a dynamic system behavior using a combination of hierarchical organizational structure and control feedback loops. Table 1 shows the summary of the findings for this criterion and highlights approaches that are passing this selection criterion. Mark “X” in Table 1 denotes whether approach satisfies the statement in the heading of the column.

Table 1 - Summary for selection criterion „Modelling support“

<table>
<thead>
<tr>
<th>Approach</th>
<th>Addresses human factors</th>
<th>Addresses organizational factors</th>
<th>Addresses linear connections</th>
<th>Addresses dynamic connections</th>
</tr>
</thead>
<tbody>
<tr>
<td>AEB</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>HERA</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Domino model</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fault tree analysis</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Root cause analysis</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Five whys</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Swiss cheese</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>SERA</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Rasmussen’s hierarchical model</td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>STAMP</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>FRAM</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

- **Classification of failures**: defines the means to identify risks. The goal is to select approaches that provide a comprehensive classification of failures for all identified socio entities. Among the approaches presented in Subsections 2.6.1 and 2.6.2 it is possible to highlight FRAM, STAMP, HERA and SERA as approaches that provide classification for identification of failure of socio entities. HERA provides the classification that addresses human and some of the organizational failures, but it is too focused on avionics domain. FRAM provides short classification of human and organizational preconditions for performance variability. However, since FRAM uses functional representation of the system, it is hard to directly associate preconditions to the human or organizational entity. STAMP is focused on the control flaws and
provides a brief general classification of failures that can arise in feedback control loops. However, the provided classification is mostly applicable to organizational entities. On the contrary, SERA provides an exhaustive classification for both human and organizational factors on multiple levels. SERA offers not only a classifications of faults, but also provides a matrix of possible interrelatedness between organizational and human faults. Table 2 shows the summary for this criterion and highlights approaches that are passing the selection criterion.

<table>
<thead>
<tr>
<th>Approach</th>
<th>Classification of human failures</th>
<th>Classification of organizational failures</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>AEB</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HERA</td>
<td>X</td>
<td>X</td>
<td>Too specific to avionics domain</td>
</tr>
<tr>
<td>Domino model</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fault tree analysis</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Root cause analysis</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Five whys</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Swiss cheese</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SERA</td>
<td>X</td>
<td>X</td>
<td>General classification of human and organizational failures</td>
</tr>
<tr>
<td>Rasmussen’s hierarchical model</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>STAMP</td>
<td></td>
<td>X</td>
<td>Classification of control failures</td>
</tr>
<tr>
<td>FRAM</td>
<td></td>
<td>X</td>
<td>Short classification of preconditions to variability</td>
</tr>
</tbody>
</table>

Based on the first criterion, that is modeling support, our attention is narrowed to SERA, Rasmussen’s hierarchical model, FRAM and STAMP since they address both human and organizational entities and are able to capture the dynamic relations between the identified entities in STS. However, based on the second criterion, which is the classification of failures, the attention is narrowed to STAMP, FRAM, HERA and SERA that offer some classification of failures for socio-entities. However, HERA is eliminated by the first criterion. Rasmussen’s approach addresses complex and dynamic behavior of STS but lacks in providing a fine classification of failures for socio-technical entities. STAMP model also addresses dynamic relations between entities in STS and provides a general classification of failures for control loops, describing failures of both human and organizational functions. Also, STAMP classification is coarse too general to be associated to only one entity. FRAM also models
dynamic system behavior but provides brief classification of faults for socio-technical entities. On the other side, it is possible to highlight SERA as an approach which provides exhaustive fine grained failure classification for human and organizational entities and using PCLs addresses dynamic relations between those entities. Using failure classification and PCLs, SERA addresses the interrelatedness of organizational faults and human faults. SERA classification of failures is based human perception of the world and human decision making process.

Considering the aforementioned facts, SERA is chosen as an approach that will guide the further work in this thesis related to identifying and modelling failure behavior for socio-technical entities. Since the SERA approach addresses humans and organizations, it is necessary to find a suitable failure behavior modelling and an analysis technique that can be applied to all identified entities in STS, including technical entities. In the Subsection 6.1.4 such technique is selected.

6.1.4. Which technique for analysis of failure behavior should be used?

This section represents a continuation of the state-of-the-art presented in Section 2.5. A research on the topic of modelling failure behavior of hardware and software components in the context of the CHESS project was conducted in [19]. Even though this thesis is defined in the context of the CONCERTO project, it is worth to explore the state-of-the-art outside CONCERTO and its heritage. In this section, techniques and tools selected in Section 2.5 are compared from the CONCERTO perspective. The existing design models of hardware and software architectures represent a good basis towards a comprehensive modelling support for socio-technical systems that can be extended to socio entities as well. When considering existing approaches, we need to choose the ones that support a component-based model-driven approach that will enable further decomposition of socio entities and also the extraction of extrafunctional properties. Among many available tools and techniques, our attention is narrowed to the component-based model driven approaches such as AltaRica, AADL, HiP-HOPS and FPTC.

From the tool support point of view, the FPTC technique steps out among other approaches because of the possibility of generation of the FTA and FMEA tables for models with cycles. According to [27], FPTC combines two traditional and widely accepted risk identification techniques (FTA and FMEA) and calculates a system level behavior based on the behavior of individual components. It supports the modelling of feedback loops present needed to model dynamic relations in socio-technical systems.

Even though AltaRica offers model based analysis of failure behavior, it does not provide a complete tool for failure behavior from the design models. It relies on other tools for generating components that can be used as inputs for AltaRica. On the other hand, Osate provides a full tool support for the modelling and analysis of failure behavior of software and hardware architectures using AADL. EAST-ADL2 also provides the modelling support, while failure behavior analysis is conducted using external tools.
In contrary to HiP-HOPS and AADL, the FPTC technique, implemented in CHESS-FLA, offers a fully automated tool support to model and analyze failure behavior. It also supports the backpropagation of analysis results to the original models, which keeps the models consistent. Thus, FPTC implemented in CHESS-FLA is chosen as a technique for the analysis of failure behavior of all entities in STS. Also, this section answers the question “Which tool supported technique can be extended to support modelling and analysis of failure behavior of socio-technical entities” since FPTC has a good tool support implemented as the CHESS-FLA plugin in the CONCERTO tool-set.

6.2. Proposed solution – Concereto-FLA

In Section 6.1, the SERA approach has been selected for identifying the socio-technical entities and their failure types. After the initial analysis of SERA in Subsection 6.1.4, human and organizational entities are identified as two entities from a high level perspective. However, in order to follow the MTO suggestions for a joint analysis of socio-technical entities, it is necessary to cover technological entities as well. Therefore, the entities identified using SERA should be combined with the CHESS-FLA modelling and analysis technique selected in Subsection 6.1.4. A possible way of combination of the SERA and CHESS-FLA techniques is an approach named CONCERTO-FLA presented in [5]. CONCERTO-FLA emerged as a result of the analysis part of this thesis. CONCERTO-FLA represents the first step towards the support for modeling of failure behavior for socio entities in the context of the CONCERTO project. However, CONCERTO-FLA covered only one human factor and its preceding organizational factor, namely “time pressure” as organizational failure and failure in “attention” as human failure. Still, the full spectrum of the SERA active failures and preconditions for all entities in the STS needs to be considered. The main contribution of this thesis represents a continuation of CONCERTO-FLA and delivers an approach named Concereto-FLA. The renaming is motivated by the need to distinguish the project name from technique name. Concereto-FLA further decomposes human and organizational entities by the means of the SERA taxonomy. The contribution is described in more details in Subsections 6.3.1, 6.3.2 and 6.3.3, which are devoted to the interpretation of the SERA active failures and preconditions in terms of the FPTC implementation in CHESS-FLA.

SERA active failures and preconditions describe in which way the human and organizational internal functionality can fail. SERA active failures and conditions of the personnel are associated to a human entity, while the preconditions and their relation with active failures can be expressed through the interconnections between entities in a system.

After thorough inspection of the SERA human active failures, it is possible to conclude that human failures can be related to two types of human functionalities: internal functionalities (such as sensory, knowledge, perception, etc.) and functionalities responsible for acting (still internal to the human, but responsible for the output action). This decomposition is motivated by human behavior. A human must process the input information in form of a sense, perception or as knowledge. Then, the internal functionality will process the inputs and form an action as a result of the stimulus on the input on human composite. The action might be
different if different internal functionality contributed to forming an action. Hence, the proposed approach is to model human beings as composite entities that contain subentities which represent their internal functionalities and actuator-like functionalities. The proposal is to model twelve SERA active failures as an internal entity of human composite entity (see Figure 11). These internal entities are named according to the SERA active failures (e.g. Attention is the name of internal entity related to attention failure).

![Figure 11 - Human composite entity with internal structure](image)

Internal entities inside the human composite entity (shown as internal functionality in Figure 11) are connected through appropriate ports to the human internal actuator entity, in this proposal named Action (as shown in Figure 11). In order to express the influences of several active failures that can occur at the same time, several internal entities representing internal functionalities of human beings can be connected to the one actuator entity.

Human entity receives inputs through input ports (port In in Figure 11). These inputs are then forwarded to the entity that represents corresponding internal functionality. Internal functionality produces an input to Action entity, which then produces the physical output to the environment through the output port (port Out in Figure 11).

After thorough inspection of the organizational preconditions for the active failures, the similar method used for the failures associated with human entity could be applied to model the organizational preconditions (see Figure 12) as well. The SERA organizational preconditions can be represented as the global organizational factors and their specializations (preconditions), modelled as interconnected sub-entities (atomic or composite) or an organizational composite component. The decomposition of the organizational entity is motivated by the SERA approach which identifies a big number of organizational factors that are contributing to the occurrence of the human failures. However, since one factor might influence other factors as well, the preconditions can be interconnected. Also, one external factor might impact more than one entity inside organization. However, the domain expert should define the valid connections between those entities.
To capture the organizational influences on human entities, the organizational and human composite, entities should be connected through appropriate ports (see Figure 13). The connection between human and organizational entities and between human and technical entities enables failure propagation between levels in the organizational hierarchy. In Figure 13, those connections are established through Out_1 – In_1 and Out_2 – In_2 connectors.

The preconditions related to the personnel such as the states of the individual, working both individually and as a team or a group can be divided onto preconditions related to the state of the individual (Physiological, Psychological, Social, Physical capability, and Personal readiness), and onto preconditions interpreted as the organizational influence (Training and Selection, and Qualification and authorization). Therefore, a proposal is to model Training and Selection, and Qualification and authorization as internal entities inside organizational composite entity. The motivation for this proposal is the fact that organization is doing a selection of its employees and is either responsible to train the personnel or to choose trained personnel for a task. Similarly, organization is responsible for qualification and authorization of their employees. Also, a proposal is to model states such as Physiological, Psychological, social, Physical capability, and Personal readiness, as internal entities to human composite entity. Therefore, these entities reflect the human state and the output of these entities should be connected to the human internal entities. The motivation for this proposal is in the fact that human state can influence internal functionalities (i.e. Sensory) and actuator entities.

To support safety analysis, all architectural elements, including human and organizational entities and their internal subentities, must have at disposal the language elements for decoration with safety-related information defined in [27] for software and hardware components. These language elements must be, available for the connectors as well since accidents in complex socio-technical systems can occur due to the external factors or the inappropriately handled interconnections between the system components. The inappropriate interconnections are not always visible during the design. Thus an overall system analysis needs to be performed to reduce the deviant connections and behavior resulted from that.
The proposal is to express human and organizational failures in form of failure modes, proposed in [6] and extended in [47]. Therefore, the SERA active failures and preconditions should be expressed using the FPTC failure modes selected in Subsection 6.1.4, which will enable analysis of failure propagation through all system components, socio and technical. The interpretation of human and organizational failures in terms of failure modes represents the main contribution of this thesis. This is further described in Section 6.3.

The expression of human and organizational failures through the failure modes results in creation of an incidence matrix of the possible combinations of human and organizational failures, as shown in Figure 13. Grey cells in Figure 13 represent the possible connection between organizational precondition and human failure. The incidence matrix depicts the possible relations and failure propagation flow between organizational subentities and human subentities. Such matrix can represent either generic or domain-specific propagation flows filled in by the domain experts. This thesis provides a possible way of combining human and organizational failures for the petroleum domain. Limited by space in this thesis, incidence matrices are provided for each human failure and organizational precondition, as shown in Subsections 6.3.1, 6.3.2, and 6.3.3. The proposed matrices should be expanded so they include all identified socio entities and their outputs.
The Concerto-FLA modelling and analysis should be supported by the CONCERTO-ML modelling language. It should allow the creation of models of systems composed of all socio-technical entities. The dependability information can be extracted from design models using model transformations recalled in Subsections 2.3.1, 2.4.3 and 2.4.4. An extension of CONCERTO-ML will allow automatic analysis and computation of dependability characteristics on a system level. Figure 14 shows an activity diagram of the proposed solution to calculate failure behavior of socio-technical systems based on the FPTC failure logic specified at a component and connector level.

![Activity diagram related to Concerto-FLA](image)

**Figure 14 - Activity diagram related to Concerto-FLA [5]**

### 6.3. Failure types that characterize the failure behavior of socio entities

The interpretation of socio-behavior in terms of the FPTC failure modes implemented in CHESS-FLA represents the first step in realization of the proposed Concerto-FLA. This section describes the main contribution of this thesis reflected in a proposal to model the human and organizational failure behavior based on the limited expertise collected in a limited time-frame for this thesis. The lack of expertise resulted that some proposals are based on assumptions which need to be revised by a domain expert before application. However, it presents a good starting solution for future improvements by domain experts and adaptation to applicable domains. This section starts with interpreting the organizational preconditions in terms of FPTC in Subsection 6.3.1 and continues with interpreting human preconditions and human active failures in terms of FPTC in Subsections 6.3.2 and 6.3.3. In Subsection 6.3.3, for each human failure, an incidence matrix of human and organizational failures is provided.

#### 6.3.1. Organizational preconditions in terms of CHESS-FLA FPTC-failure types

Organizational preconditions identified in the SERA classification and recalled in Subsection 2.6.2 are represented as sub-components of the organizational composite component. The contribution of the thesis is also reflected in defining the failure modes for each organizational precondition in terms of CHESS-FLA FPTC-failure types. The following list describes how these preconditions are modelled and expressed in terms of CHESS-FLA FPTC-failure types:
• **Time pressure** - Time pressure is modelled as a component named *TimePressure* that can produce `valueCoarse` (when inappropriate amount of time pressure is detectable by humans) and `valueSubtle` (when inappropriate amount of time pressure is not detectable by humans). According to description of time pressure found in SERA, time pressure is unwanted and wrong organizational behavior since it occurs when workers cannot handle extreme task tempo. As such, it cannot be expected by humans and *TimePressure* component cannot produce timing and provision-related FPTC-failure types;

• **Objectives** - Objectives precondition is represented as an organizational subcomponent named *Objectives*, and it can produce a `valueSubtle` and `valueCoarse` failure denoting the faulty objectives, and an *omission* denoting the lack of objectives. Time related failures are not possible to happen since objectives are defined prior to task execution and cannot be late or early;

• **Equipment** - Equipment precondition is represented as an organizational subcomponent named *Equipment* which can produce `valueSubtle` and `valueCoarse` failures denoting the faulty equipment, and an *omission* denoting the lack of equipment needed for completing a task. A time related failure cannot occur since the equipment cannot be provided early, while late provision of equipment is covered by the omission failure, meaning the task has started with the lack of equipment;

• **Workspace** - Workspace is represented with a component named *Workspace* that can produce `valueSubtle` and `valueCoarse` denoting the inappropriate layout of the workspace for a specific activity. Since the workspace is always present and valuated quantitatively, the associated component cannot produce failures related to provisioning or timing;

• **Environment** – Since the environment is always present, it cannot produce time-related or provision related failures. Thus, the environment is modelled as an organizational subcomponent named *Environment* that can produce `valueSubtle` and `valueCoarse` failures;

• **Forming intent** - Forming intent is represented as a component named *FormingIntent* that can produce `valueSubtle` and `valueCoarse` denoting the inappropriately formed intent for the activity. The time and provision related failures cannot be associated to this component since the intent for the task is always present;

• **Communicating intent** - Communicating intent is represented as a component named *CommunicatingIntent* that can produce `valueSubtle`, `valueCoarse`, `late`, and *omission*. Value-related failures denote the partially transferred information about the intent, failure *late* denotes the intent that is communicated lately, while *omission* denotes the lack of the communication at all. Failures *early* and *commission* cannot occur since the intent cannot be communicated when unexpected and early information is always useful;

• **Monitoring and supervision** - Monitoring and supervision are represented as a component named *MonitoringAndSupervision* that can produce all types of the FPTC failures except *early*, where value-related failures represent the inadequate supervision and feedback activity, time-related failures describe the late or delayed feedback and
provision-related failures describes the absence of these activities. Early FPTC failure is not considered as a failure since early feedback cannot cause any failure;

- **Mission** - Mission is modelled as the component named Mission that can produce valueSubtle, valueCoarse and omission failures. Value related failures represent the unclear and inappropriate mission definition, while omission denotes the mission that is not approved. Time related failures cannot occur since it is not possible to have early of late mission;

- **Provision of resources** - Provision of resources is modelled as a component named ProvisionOfResources and can produce valueSubtle, valueCoarse, omission, commission, and late. An early provision of the resources is not considered as a fault;

- **Rules and regulations** - Rules and regulations are modelled as a component named RulesAndRegulations and can produce valueSubtle, valueCoarse, commission and omission. Value-related failures denote the inappropriate rules and rules inconsistent with the mission requirements; commission denotes unexpected regulations when the organization is not prepared, while omission represents the lack of rules and regulations. Time related failures cannot occur since the rules and regulations cannot be described as early or late. The RulesAndRegulations component can be connected to the external body to include the external regulatory body usually present in safety-critical domains;

- **Organizational process and practice** - The component that represents Organizational process and practice is named Process and can produce valueSubtle, valueCoarse, and an omission. Value-related failures describe the faulty (inappropriate, ambiguous, etc.) processes and procedures, while an omission describes the lack of the processes and practices. Time related failures cannot occur since processes and practices are always defined;

- **Organizational climate** - Organizational climate is modelled as a component named Climate and can produce valueSubtle and valueCoarse failures. Value-related failures describe the inefficient structure, control and monitoring chain, inappropriate policies and presence of the unspoken rules that cause the acting against the official rules. The time and provision related failures cannot occur since the organizational climate is always present and defined;

- **Oversight** - The oversight is represented with the component named Oversight that can produce valueSubtle, valueCoarse, and an omission. Value-related failures represent the faulty management procedures while an omission failure represents the lack of those. Time related failures cannot occur since learning mechanisms cannot be described as early or late.

6.3.2. Human preconditions in terms of CHESS-FLA FPTC-failure types

The human preconditions identified in the SERA classification and recalled in Subsection 2.6.2 are represented as sub-components of the human composite component. As mentioned in Section 6.2, Training and selection, and Qualification and authorization should be modelled as internal entities of the organizational composite entity, while the rest of the preconditions are related to human states and are modelled as internal functionalities of human composite
entity. However, according to SERA, human states cannot be impacted by organization, but the state of the human can impact both, action and human internal functionality. The following list describes how these preconditions are modelled and expressed in terms of FPTC:

- **Physiological state** – In Concerto-FLA, human physiological state is modelled as a component named *PhysiologicalState*. It can produce *valueSubtle* and *valueCoarse* on its outputs denoting the wrong physiological state of a human that performs the task.

- **Psychological state** - In Concerto-FLA, human psychological state is modelled as a component named *PsychologicalState*. It can produce *valueSubtle* and *valueCoarse* on its outputs denoting the wrong psychological state of a human that performs the task.

- **Social factors** - In Concerto-FLA, social factors are modelled as a component named *SocialFactors*. It can produce *valueSubtle* and *valueCoarse* denoting the wrong social factors present in a team or a group that affect the individual or a group.

- **Physical capability** - In Concerto-FLA, human physical capability is modelled as a component named *PhysicalCapability*. It can produce *omission* denoting the case in which human physical ability is not appropriate for completing the task.

- **Personnel readiness** - In Concerto-FLA, human readiness is modelled as a component named *Readiness*. It can produce *omission* denoting the case in which human readiness is not delivered for completing the task.

- **Training and selection** - In Concerto-FLA, human training and selection is modelled as a component named *TrainingAndSelection*. Since it relates to the lack of training and knowledge this component can produce *omission* denoting the lack of training and knowledge for completing the task.

- **Qualification and authorization** - In Concerto-FLA, qualification and authorisation are modelled as a component named *QualificationAndAuthorisation*. It can produce an *omission* which denotes the lack of proper qualification or authorization for performing a task.

6.3.3. Human failures in terms of CHESS-FLA FPTC-failure types

The human active failures identified in the SERA classification and recalled in Subsection 2.6.2 are represented as sub-components of the human composite component. The following list describes how these are modelled and expressed in terms of FPTC, which is a part of the novel contribution of this thesis. For each identified component, an incidence matrix is proposed based on the SERA mappings of active failures and preconditions (see Appendix A), and also based on the FPTC failures that each component can produce. The preconditions are already expressed in Subsections 6.3.1 and 6.3.2.

- **Sensory Failure** - In Concerto-FLA, human sensory capability is modelled as a component named *Sensory* that can produce *valueSubtle*, *valueCoarse*, and *omission* FPTC failures. When a human does not have the sensory capability required by task, *Sensory* fails with omission that is forwarded to the *Action* component. Value-related FPTC failures occur when human sensory ability is degraded and a human can or cannot be aware of the fault and thus *Sensory* component fails with the *valueSubtle*
and valueCoarse respectively. Human sensory ability cannot be described using the time-related FPTC failures since sensing information cannot be forwarded early or late, but instantaneously. SERA identified the following organizational influences that can cause human sensory failure: Objectives, Equipment, Environment, Monitoring and supervision, Process and Oversight. An incidence matrix showing sensory failure and its preconditions expressed using the FPTC failures is shown in Figure 15.

![Figure 15 - Incidence matrix of Sensory failure and its preconditions in terms of FPTC failures](image)

- **Intent Failure** - In Concerto-FLA, intent failure is modelled as a component named Intent. The risky or inappropriate goal is represented as valueSubtle (non-violation) and valueCoarse (violation) failures. The FPTC failures related to timing and provisioning cannot appear at Intent output since an intention is formulated in the human mind before the action is performed. SERA identified the following organizational influences that can cause human intent failure: Time pressure, Objectives, Communicating intent, Provision of resources, Monitoring and supervision, Mission, Process, Climate, Rules and regulations, and Oversight. Figure 16 shows the incidence matrix of intent failure with its preconditions expressed using the FPTC failures.

![Figure 16 - Incidence matrix of Intent failure and its preconditions in terms of FPTC failures](image)

- **Attention failure** - In Concerto-FLA, human attention capability is represented as the Attention component and it can produce an omission failure. Omission at the output denotes a failure of human attention capability to deliver the available information. In
order to not fail the human needs to attend all present information, thus value and time related FPTC failures are not possible to happen. SERA identified the following organizational influences that can cause human attention failure: Time pressure, Equipment, Environment, Provision of resources, Monitoring and supervision, Oversight, Mission, Process, and Climate. Figure 17 shows the incidence matrix of attention failure with the possible preconditions expressed using the FPTC failures.

Figure 17 - Incidence matrix of Attention failure and its preconditions in terms of FPTC failures

- **Knowledge (perception) failure** - In Concerto-FLA, this failure is modelled as a component named KnowledgePerception. It can produce omission which denotes the lack of knowledge to perceive the situation. The expected information is not provided and the situation cannot be perceived. If correct information is present in the knowledge base, it will be forwarded to Action, thus the time and value related FPTC failure cannot appear at the output. SERA identified the following organizational influences that can cause human knowledge (perception) failure: Objectives, Forming intent, Communicating intent, Provision of resources, Monitoring and supervision, Oversight, Mission, and Process. Figure 18 shows the incidence matrix of Knowledge (Perception) failure and its possible preconditions expressed using the FPTC failures.

Figure 18 - Incidence matrix of Knowledge (Perception) failure and its preconditions in terms of FPTC failures

- **Perception Failure** - In Concerto-FLA, perception failure is modelled as a Perception component, and it can produce ValueSubtle as an output. ValueSubtle denotes the wrong state of the world that is perceived and propagated to other human logic components. A human is not aware of the wrong perception and Perception component cannot produce ValueCoarse. Perception is always present in human logic and therefore Perception component cannot fail with the time or provision related
FPTC failures. SERA identified the following organizational influences that can cause human perception failure: Equipment, Environment, Monitoring and supervision, Oversight, Process, Climate. Figure 19 shows the incidence matrix of perception failure with its possible preconditions expressed using the FPTC failures.

![Figure 19 - Incidence matrix of Perception failure and its preconditions in terms of FPTC failures](image1)

- **Communication/information failure** - In CONCERTO-FLA, this failure is modelled as a Communication component and it can produce omission, late, valueSubtle and valueCoarse FPTC failures at the output ports. Omission at the output denotes the case when no information is received. Late occurs when a communication channel is slow and provides the information late. The value related FPTC failures denote the incorrect information (valueSubtle) but a human can recognize the channel malfunction (valueCoarse). SERA identified the following organizational influences that can cause human communication failure: Equipment, Environment, Monitoring and supervision, Oversight, Process, and Climate. Figure 20 shows the incidence matrix of the communication failure with its possible preconditions expressed using the FPTC failures.

![Figure 20 - Incidence matrix of Communication failure and its preconditions in terms of FPTC failures](image2)

- **Time Management Failure** - In Concerto-FLA, this failure is modelled as a TimeManagement component that can produce valueSubtle and valueCoarse FPTC
failures. Value related failures denote the inadequate time management strategy used by a human to complete the task. The TimeManagement component cannot fail with time and provision related FPTC failures since some time management strategy is always present and used. SERA identified the following organizational influences that can cause human time management failure: Time pressure, Provision of resources, Monitoring and supervision, Oversight, Mission. An incidence matrix for Time management failure and its possible preconditions expressed using the FPTC failures is shown in Figure 21.

<table>
<thead>
<tr>
<th>FPTC Failure</th>
<th>Time Pressure</th>
<th>Provision Of Resources</th>
<th>Monitoring And Supervision</th>
<th>Oversight</th>
<th>Mission</th>
<th>Training And Selection</th>
</tr>
</thead>
<tbody>
<tr>
<td>noFailure</td>
<td>valueCoarse</td>
<td>valueCoarse</td>
<td>omission</td>
<td>omission</td>
<td>omission</td>
<td>omission</td>
</tr>
<tr>
<td>valueCoarse</td>
<td>omission</td>
<td>omission</td>
<td>omission</td>
<td>omission</td>
<td>omission</td>
<td>omission</td>
</tr>
<tr>
<td>omission</td>
<td>omission</td>
<td>omission</td>
<td>omission</td>
<td>omission</td>
<td>omission</td>
<td>omission</td>
</tr>
<tr>
<td>valueSubtle</td>
<td>omission</td>
<td>omission</td>
<td>omission</td>
<td>omission</td>
<td>omission</td>
<td>omission</td>
</tr>
</tbody>
</table>

Figure 21 - Incidence matrix of Time management failure and its preconditions in terms of FPTC failures

- **Knowledge (Decision) Failure** - In Concerto-FLA, this failure is modelled as a component named KnowledgeDecision. It can produce omission which denotes the lack of knowledge to perceive the situation. The expected information is not provided and the correct decision cannot be made. If correct information is present in the knowledge base it will be forwarded to Action, thus the time and value related FPTC failure cannot appear at the output. SERA identified the following organizational influences that can cause human knowledge (decision) failure: Mission, Monitoring and supervision, Process, Oversight. Figure 22 shows the incidence matrix of Knowledge (Perception) failure and its possible preconditions expressed using the FPTC failures.

<table>
<thead>
<tr>
<th>FPTC Failure</th>
<th>Mission</th>
<th>Monitoring And Supervision</th>
<th>Process</th>
<th>Oversight</th>
<th>Training And Selection</th>
</tr>
</thead>
<tbody>
<tr>
<td>noFailure</td>
<td>valueCoarse</td>
<td>omission</td>
<td>omission</td>
<td>omission</td>
<td>omission</td>
</tr>
<tr>
<td>valueCoarse</td>
<td>omission</td>
<td>omission</td>
<td>omission</td>
<td>omission</td>
<td>omission</td>
</tr>
<tr>
<td>omission</td>
<td>omission</td>
<td>omission</td>
<td>omission</td>
<td>omission</td>
<td>omission</td>
</tr>
<tr>
<td>valueSubtle</td>
<td>omission</td>
<td>omission</td>
<td>omission</td>
<td>omission</td>
<td>omission</td>
</tr>
</tbody>
</table>

Figure 22 - Incidence matrix of Knowledge (Decision) failure and its preconditions in terms of FPTC failures
**Ability to respond failure** - In Concerto-FLA, this failure is represented as a component named *Response* and it can produce *omission*, *valueCoarse* and *valueSubtle* FPTC failures on output port. *Omission* at the output denotes the lack of the ability to provide a correct response. Insufficient ability is represented with value related FPTC failures. A human can or cannot be aware of the inabilities and thus a *Sensory* component can also fail with *valueSubtle* and *valueCoarse*. SERA identified the following organizational influences that can cause a human ability to respond failure: Objectives, Equipment, Workspace, Environment, Provision of resources, Monitoring and supervision, Process, Oversight. Figure 23 shows the incidence matrix of ability to respond failure and its possible preconditions expressed using the FPTC failures.

![Figure 23 - Incidence matrix of Ability to respond failure and its preconditions in terms of FPTC failures](image)

**Action selection failure** - In Concerto-FLA, this failure is represented as a component named *Action Selection* and it can fail with *valueSubtle* FPTC failure. *ValueSubtle* represents a wrongly selected action to perform a task. The human is not aware of the wrong action and thus *valueCoarse* is not possible as an output. Action selection will always produce output since ‘no action’ is also considered as an output, thus time and provision related FPTC failures cannot occur. SERA identified the following organizational influences that can cause human action selection failure: Time pressure, Equipment, Mission, Provision of resources, Monitoring and supervision, Oversight. The incidence table of action selection failure with its possible preconditions expressed using the FPTC failures is shown in Figure 24.

![Figure 24 - Incidence matrix of Action selection failure and its preconditions in terms of FPTC failure](image)
• **Feedback failure** - In Concerto-FLA, feedback failure is represented as a component named *Feedback*. Since feedback failure means lack of error correction, the component *Feedback* can fail with omission value *Coarse*, *valueSubtle*, *early* and *late* on its output. *Omission* denotes the complete lack of feedback, value related FPTC failures denote the incorrect error correcting data, while time related FPTC failures denote the early or late feedback. Early feedback can be ignored if a human is busy with other action. SERA identified the following organizational influences that can cause human feedback failure: Time pressure, Equipment, Environment, Provision of resources, Monitoring and supervision, Oversight, Mission. The incidence table of this failure with its possible preconditions expressed using the FPTC failures is shown in Figure 25.

![Incidence matrix of Feedback failure and its preconditions in terms of FPTC failure](image)

**Figure 25** - Incidence matrix of Feedback failure and its preconditions in terms of FPTC failure

• **Slips** - In Concerto-FLA, slips lapses and misses are represented as a component named *Slips*. It represents a trigger in human mind that causes the *Action* component to fail. A slip can be described as an action with a wrong result or value. Slips occur unintentionally and unexpectedly and thus *Slips* can fail with *commission* FPTC failure. Component *Slip* cannot produce a wrong value since any value passed to *Action* component will cause wrong action on the output. Component *Slips* cannot produce time related FPTC failures since slips and lapses are not expected by a human and information cannot come early or late. SERA identified the following organizational influences that can cause human slips and lapses: Equipment, Rules and regulation, Process and Oversight. The incidence table of this failure with its possible preconditions expressed using the FPTC failures is shown in Figure 26.
6.4. **Automated fine grained analysis of failure behavior of socio-technical entities**

As mentioned in Section 2.4, the CONCERTO framework provides the dependability view that allows various analysis types. It allows an execution of safety analysis techniques using the CHESS-FLA plugin. At the moment, CHESS-FLA allows the FPTC and FIT_SA analysis to be performed on software and hardware components. [27] As it is proposed in the solution in Section 6.2, the existing support for the failure behavior modelling and analysis of hardware and software entities can be extended on other entities such as socio entities, their internal components and connectors. Since CHESS-FLA uses the M2T transformations to extract dependability related information from models, it is necessary to update these transformations to support the new entity types. The backpropagation of analysis results to the original model also needs to be updated to support social entities. The next section describes the proposed changes to CONCERTO-ML that will allow integration of Concerto-FLA within the CONCERTO tool-set.

6.4.1. **Proposal for extensions in CONCERTO-ML**

As mentioned in Section 6.2, an integration of the proposed solution into the CONCERTO tool-set requires changes of the CONCERTO component model and the underlying CONCERTO-ML. The modelling language has to be extended to support the modelling of new types of entities identified in Section 6.1 and 6.3. This change would enable the language constructs needed for the modelling and analysis of failure behavior of socio-technical systems.

Therefore, a proposal for extension of CONCERTO-ML should include the following changes:

- Enable support for modelling “Human”, “Organizational” and “Technical” entities;
- Enable modelling of “Human” type as a composite component with the following internal components representing human active failures: Intent, Attention, Sensory,
KnowledgePerception, Perception, Communication/Information, TimeManagement, KnowledgeDecision, Response, ActionSelection, SlipLapsesMode, FeedbackFailure;

- Enable modelling of “Human” type as a composite component with the following internal components representing human preconditions: Physiological, Psychological, Social, PhysicalCapability, Readiness, TrainingAndSelection, QualificationAndAuthorisation;

- Enable modelling of “Organizational” type as composite with the following internal components: TimePressure, Objectives, Equipment, Workspace, Environment, FormingIntent, CommunicatingIntent, MonitoringAndSupervision, Mission, ProvisionOfResources, RulesAndRegulations, Process, Climate, Oversight.

Additional constraints should be possible to set according to the domain-specific incidence matrix of active failures and preconditions filled by the domain experts.

6.5. Integration of the proposed solution into the CONCERTO tool-set

The changes proposed in Section 6.4 have been partially implemented by the CONCERTO engineers. It is worth to clarify that among the proposed changes CONCERTO engineers implemented only support for human, organizational and human entities. Also, a new system view named SystemView has been added. It allows the modelling of socio-technical systems as compositional block entities in the block diagram model. Socio-technical entities can be modelled as blocks with stereotypes <<Human>>, <<Organizational>>, and <<Technical>>. Once the system is modelled, it is necessary to generate an instance of the system which will be used as input to the analysis tool.

In order to enable failure behavior analysis for new language constructs, it is necessary to update the existing CHESS-FLA. Subsection 6.5.1 describes how the existing model transformation that extracts dependability related information from system models is updated to provide input in a format expected by failure logic analysis. Once the analysis is preformed, the analysis results should be backpropagated to the original model. Therefore, Subsection 6.5.2 describes how a backpropagation mechanism is updated to work with the new system view and model entities. It is necessary to highlight that at the moment of delivering this thesis report, the implementation presented in this section has evolved towards support for state-based analysis [3].

6.5.1. Updates of M2T transformation

The M2T transformation from block model to xml is updated by adding a new Acceleo module called GenerateXML_SYS. That module defines XML_SYS template. In order to call this a template, the code shown in Figure 27 has been added to the GenerateFPTC module. This code checks whether the selected platform belongs to a system view (modelSystemView) and calls XML_SYS transformation. The main difference between the Concerto-FLA and CHESS-FLA lies in the entity type used for modeling the system. In the block diagram, system is modelled as a composite block which is a type derived from type Class and not from Component type as in CHESS-FLA.
The XML_SYS template contains a code for transformation of block diagrams to xml formatted input for failure logic analysis.

- **Input and output ports**

The code in Figure 28 shows the transformation code for extracting information about input and output ports of the top-level composite block (system). Input ports of the top-level block are annotated with a comment with stereotype <<FPTCSpecification>> which allows the specification of input failure type. Input failures are then stored into the inputValues attribute as a comma separated list.

- **Transformation rules**

The code shown in Figure 29 represents the code for extracting transformation rules as one string. For loops in lines 1 - 4 iterate through all instances and their slots until the line 5. The if condition in line 5 checks whether the instance corresponds to the component whose transformation rules are currently processed. Once the condition is met, for loop in line 6 iterates through all properties of the instance and choses the ones with applies FPTC stereotype. Let block in line 8 and 9 extracts all rules associated to the instance as one string and store it into variable rule. Since all rules stored in variable rule are separated by a semicolon, variable rule is further passed to getTransformationRules template for detailed parsing.
Template `getTransformationRules` is also updated and it looks as shown in Figure 30. It separates string on many substrings based on ‘;’ character. The update for this function is the `if` block which checks the validity of the rule by checking the presence of ‘->’ and ‘.’ characters.

```
[template public getTransformationRules(rule: String)]
<TransformationRules>
[for eoneRule: String | rule.tokenize(';'))]
  [if (eoneRule <> null) and eoneRule.contains('->') and eoneRule.contains('.'))]
    <Expression lhsPattern="oneRule.strip('->',0).trim()" rhsPattern="oneRule.strip('->',1).trim()" />
  [/if]
[/for]
[/template]
```

A template for extracting information about connections has also been adapted (see Figure 31). This template extracts the information about source entity stored in `srcOwnerName`, source port stored in `srcName`, destination entity stored in `destOwnerName` and destination port stored in `destName`. The updated code is located in the `else` block which starts from line 11 in Figure 31. The update of `getConnectionsInfo` was necessary since the destination entity for UML blocks was not written directly in the connector. Instead, the UML block itself holds a reference to defining feature which gives the name of block instead of block instance. For that reason, it was necessary to search through all block instances and find the name of the destination block instance. `For` loop in line 12 on Figure 31 iterates over destination ports and in line 16 matches the port’s parent name with instance names of all blocks. Once the matching is found the block instance name will be printed as `destOwnerName` (line 17 in Figure 31).
6.5.2. Update of backpropagation of analysis results

After a successfully performed analysis, Concerto-FLA propagates the back analysis results into the initial model. During the backpropagation, each output port is associated to one annotated comment with the FPTCSpecification stereotype. The analysis result is stored in failure attribute of the FPTCSpecification stereotype. In order to remain backwards compatible with the support for analysis of software and hardware components, some variable names kept the name components.

Before starting a backpropagation, a SystemView is fetched (see line 8 in Figure 32) and all elements in that view are stored into a sysPackList list as shown in Figure 32. After that, the code iterates through a list in order to find a system composite block stored in variable component. Found element is converted to UML class (see line 14 in Figure 32) and it is forwarded to the PropagateBackCompBlock function which backpropagates analysis results to the block. If the block is composite, the function PropagateBackCompBlock is called for each sub-entity. If the entity is composite, it is added to the queue which is further processed in if block starting at line 26 in Figure 32. For each sub-entity stored in _compositeComponents queue a function backPropagation_subComponentBlock is called. This function assigns the analysis results to the output ports of the block.
Figure 32 - Code for calling backpropagation function

Function `PropagateBackCompBlock` iterates through all ports for given component and calls function `GetCommentBlock` to retrieve existing comments or create a new one if there is no such comment associated to the port. Once the comment is retrieved, the function retrieves the `FPTCSpecification` stereotype and writes the failures from analysis results.

Function `GetCommentBlock` (see Figure 33) searches through all existing comments associated with block and extracts `partWithPort` from the `FPTCSpecification` stereotype. This value is further used to validate and match a port name with the port specified in the `FPTCSpecification` stereotype. If the existing comment is not found, a new one is created by calling the `CreateCommentBlock` function.
The Chapter 7 shows how a developed Concerto-FLA can be used on an example from the petroleum domain presented in Chapter 3.

7. Case study

This chapter shows the modelling process of the example system presented in Chapter 3 using the Concerto-FLA plugin. In Section 8.1 system building blocks and their relations are identified, while in Section 8.2 the system model is enriched with failure behavior information and a failure behavior analysis is performed. Section 8.3 provides a discussion on the specific application of Concerto-FLA.

7.1. Modelling of the case study

In this section, system building blocks of the example presented in Chapter 3 are identified and modelled. A system is modelled as a block named GasDetectionSystem, as shown in Figure 34. A system is modelled with two input ports, namely IP1 and IP2, and one output port namely OP1. Port IP1 denotes the occurrence of the gas leak on the rig. Port IP2 denotes the inputs of regulations from the authority. Port OP1 provides the output of the system and provides information on whether the gas leak has been detected or not.

In the example system described in Chapter 3, it is possible to identify all three types of composite socio-technical entities. Auto gas detection can be modelled as a technical block.
named *AutoGasDetection*, while manual gas detection can be modelled as a human block named *ManualGasDetection*. However, a human acts in the environment created by an organization. Thus, besides human and technical, organizational factor is identified and modelled as the *Organization* block.

In order to combine the results of *AutoGasDetection* and *ManualGasDetection*, one additional block named *GasDetection* is added. *GasDetection* provides the output of the system and implements the failure logic inputs from *AutoGasDetection* and *ManualGasDetection*. Block *organization* is connected through appropriate ports to *manualGasDetection*, denoting the organizational influence on a human block. Connections between these blocks are modeled according to the provided matrices in Section 6.3. The meaning of these connections for this case study is further described when modelling the internal structure of these blocks in Subsections 7.1.1 and 7.1.2.

![Figure 34 - Identified and modelled building blocks of GasDetectionSystem](image)

**7.1.1. Modelling internal structure organizational composite block**

The organizational block is represented as a composite entity and consists of several blocks that represent organizational factors. It is modelled with an output port for each identified internal block. The identification and modelling of these influences is described below and the result is shown in Figure 35.

Lack of training is considered as a fault of the organization. It is the responsibility of the organization to either hire trained personnel, or to provide in-house training. Therefore, a lack of training is modelled as a block named *training* inside the composite *organization* block.
Working procedures are established and provided by the organization and thus are modelled as a block named procedures inside the composite Organization block.

Time pressure is a phenomenon that occurs due to a fast tempo of the working procedures. It is set by the managerial level and therefore it is modelled as a block named timePressure inside the composite Organization block.

The number of people on the rig is defined by the organizational rules but also by external factors such as illness. Looking from a perspective of a team that is performing a manual gas detection, the more people are available/provided, the more efficiently the work will be conducted. Therefore, the number of people can be represented as human resource and modelled as a block named staffing inside the composite Organization block.

7.1.2. Modelling internal structure human composite block

The modelling of human internal blocks is defined by the internal structure of organizational block. Human composite manualGasDetection is modelled with input ports that correspond to the organizational output port (see Figure 36). Also, manualGasDetection is connected to the system input port IP2 to detect a gas leak.

A lack of training is manifested as a lack of knowledge in human block. It is modelled as a block named knowledgePerception.

A lack of procedures can cause an application of improper procedure for detecting gas. This is represented as an action selection failure and it is modelled as the block named actionSelection inside the composite manualGasDetection block.

Time pressure will most likely prevent the personnel from finishing tasks in specified time and thus a human will not be able to apply a proper time management technique. Therefore, time management failure is modelled as the timeManagement block inside the composite manualGasDetection block.

A reduced number of the people on a rig decreases the sensory capability of the team to sense the gas. A reduced sensory capability is then modelled as a block named Sensory inside the composite manualGasDetection block.

The identified components, TimeManagement, Sensory, ActionSelection, and Knowledge(Perception) are connected to IP5 that denotes a gas leak occurrence, and also to the appropriate input ports IP1, IP2, IP3, and IP4 respectively. The outputs of these blocks are then forwarded to the block named Action. Therefore, the Action block has four input ports that are connected to the previously identified blocks in a human composite. The Action block provides output through the output port OP1 that is propagated further to the output of the ManualGasDetection output port OP1.
7.2. Modelling of failure behavior and automated failure behavior analysis

In the specification and analysis of failure behavior of GasDetectionSystem, the following two scenarios are interesting for this thesis. Scenario 1 is related to the case when there is only time pressure present as the organizational factor. Scenario 2 is related when there is no time pressure, but there is a lack of training, lack of procedures and also when there are few persons on the rig. In both scenarios, we assume that automatic gas detection fails with omission, and thus a gas leak can be detected by manual inspection only. Blocks in the system model have been decorated with failure behavior information according to the two scenarios as shown in Figure 35 and Figure 36. The FPTC rules are assigned to each block according to the incidence matrices defined for each type of entities (see Section 6.3). Note the applied stereotype <<organizational>> and <<human>> assigned to the block in Figure 35 and Figure 36. The current implementation of CONCERTO-ML supports assigning of these stereotypes only to block types, but not to part types. Due to that, on system level, only associated part name reveals the possible stereotype assigned on block level.

![Figure 35 - Internal structure and transformation rules of Organization block](image-url)
Concerto-FLA allows engineers to perform an automated failure behavior analysis of socio-technical systems. However, the analysis is explained in details in the following two scenarios.

- **Scenario 1**

As mentioned in the previous paragraph, this scenario represents the case when there is only time pressure present as the organizational factor, and automatic gas detection fails with omission. The failure behavior modelling and analysis results for this scenario are shown in Figure 37. In this scenario, input port IP2 of \textit{GasDetectionSystem} receives \textit{noFailure} input failure, denoting no failure from the regulatory, while IP1 receives \textit{valueCoarse} which denotes the dangerous gas leak. Therefore, \textit{noFailure} from port IP2 of \textit{GasDetectionSystem} is propagated on input port IP1 of the \textit{organization} composite block. This is further propagated to all internal blocks in the \textit{organization} composite block. Since all organizational blocks have \textit{noFailure} at their input ports, the first rule is activated, as shown in Figure 35. However, \textit{timePressure} block will produce \textit{valueSubtle} on output port OP1, denoted the high time pressure from organization. Other blocks inside \textit{organization}, namely \textit{staffing}, \textit{procedures}, and \textit{training}, will propagate \textit{noFailure} denoting normal behavior. These output failures are further propagated to \textit{manualGasDetection} through the corresponding ports and connectors.

Failures from output ports of \textit{timePressure}, \textit{resources}, \textit{procedures} and \textit{training} blocks are then propagated to input ports IP1, IP2, IP3, and IP4 of \textit{manualGasDetection} respectively.
Port IP5 of `manualGasDetection` receives `valueCoarse` and propagates it to IP2 ports of `timePressure`, `resources`, `procedures` and `training`. The failure type `valueSubtle` received on IP1 of `manualGasDetection`, denoting the presence of time pressure, is then propagated to the `timeManagement` block, and it activates the second rule. Thus, `timeManagement` will propagate further `valueSubtle` to `Action` block which means that a human could not handle the high task tempo and fails to manage time for the safe detection of gas leakage. The remaining blocks receive `noFailure` at their input ports which triggers the first transformation rule and propagates `noFailure` further to the `Action` block.

`Action` block receives failure types and matches input values with the first transformation rule. The first rule is activated and `Action` block will further propagate `late` FPTC failure to `manualGasDetection` OP1 port which denotes a late gas detection.

`ValueCoarse` from IP5 of `GasDetectionSystem` is also propagated to `autoGasDetection` and triggers the first transformations rule. This rule will transform `valueCoarse` at the input port to an `omission` on the output port OP1, which describes the failing sensors that are unable to detect a gas leak.

`Omission` from `autoGasDetection`, and `late` from `manualGasDetection` are further propagated to input ports IP1 and IP2 of the `gasDetection` respectively. This will trigger the first rule and as a result propagate failure `late` to the system output. `Late` at system output means a late gas leakage detection, which can cause an accident if the permission for welding in a critical area is already issued.

![Figure 37 - Results of failure behavior analysis using Concerto-FLA in Scenario 1](image-url)


- **Scenario 2**

As mentioned earlier in Section 7.2, this scenario represents the case in which there is no time pressure, but there is a lack of training, lack of procedures, few persons on the rig, and the automatic gas detection fails with omission. The failure behavior modelling and analysis results for this scenario are shown in Figure 38. Similarly to the scenario 1, in scenario 2 input port IP2 of `GasDetectionSystem` receives a `valueSubtle` FPTC failure, denoting a failure of regulatory to provide proper directions for the safe working procedures. Also, port IP1 receives `valueCoarse` which denotes the dangerous gas leak. Therefore, `valueSubtle` from port IP2 of `GasDetectionSystem` is propagated on input port IP1 of the `organization` composite block.

Since all organizational blocks received `valueSubtle` on their input ports, the second rule is activated, as shown in Figure 38. This time, `timePressure` block will produce `noFailure` on the output port OP1, denoting the absence of time pressure from organization. In this scenario, `staffing` produces `valueSubtle`, `training` produces `omission`, denoting a lack of training, while `procedures` produces `omission` denoting a lack of proper procedures. These output failures are further propagated to `manualGasDetection` through appropriate ports.

The failure type `noFailure` received on IP1 of `manualGasDetection`, denoting presence of time pressure, is then propagated to the `timeManagement` block, and it activates the second rule. Therefore, `timeManagement` will propagate further `noFailure` to `Action` block. Sensory receives `valueSubtle` on IP1 and `valueCoarse` on IP2 which triggers the second rule and propagates further an `omission`, denoting an omission to sense a gas leak by smelling. Because of the lack of training, block `knowledge` receives an `omission` which triggers the second rule and propagates an `omission` to the `Action` component. This denotes the lack of knowledge to understand information that a human is receiving. Failure `omission` on input port of `actionSelection` triggers the second rule which further propagates `valueSubtle`. This denotes the improper selection of the action due to a lack of procedures for gas detection.

The `Action` block receives failure types on input ports and triggers the second transformation rule. The `Action` block will further propagate an `omission` failure to `manualGasDetection` OP1 port which denotes the complete failure to detect a gas leak.

Since we have an `omission` of both, `autoGasDetection` and `manualGasDetection`, the `gasDetection` block triggers the second transformation rule and generates an `omission` on the system output denoting a complete system failure to detect a gas leak. An accident may occur if the permission for welding in a critical area is issued.
Figure 38 - Results of failure behavior analysis using Concerto-FLA in Scenario 2

7.3. Discussion

In this chapter it has been demonstrated an application of the proposed Concerto-FLA tool on an example from the petroleum industry. An example covers two scenarios which involved the interaction between human, technology, organization and environment.

Scenario 1 showed how the time pressure on a team can be propagated further in the system and transform into a failure in human time management function and might easily delay delivering of crucial information about gas leakage. Scenario 2 showed how external factors such as absence of safety regulations, imposed by regulatory body, might impact behavior of socio-technical systems. Without safety regulations imposed, an organization might easily stimulate conditions for failure occurrence without even noticing it. In Scenario 2, due to small number of people on the rig, inappropriate procedures and inexperienced personnel, a team failed to smell a gas, did not have enough knowledge to apply professional techniques and in the end made wrong decision. It is important to highlight that a lack of safety regulations is not the only root cause for the failure. The Concerto-FLA analysis identified the combinations of failures at different levels of abstraction.

In both presented scenarios in this chapter, the Concerto-FLA provided sufficient support of socio-technical concepts for risk identification and modeling failure behavior of all identified entities. An example covered all three high-level socio-technical entities such as man, technology and human. Decomposition of socio entities allows system architects to discover all aspects of human, technological, organizational, and external (e.g. regulations) factors that might contribute to the occurrence of the failure at the system output. Moreover, it supports
joint failure behavior analysis of MTO entities, and goes even further by addressing the interrelatedness of their internal functionalities. The proposed Concerto-FLA supports modelling of the hierarchical as well as flat architectures in a manner known as “divide and conquer”. Layer or subsystems can be analyzed separately or mutually at the system level.

Concerto-FLA is rather exhaustive in providing the means for identification and modelling the whole spectrum of socio-technical behavior. Simple changing of the system inputs can produce many use case scenarios and reveal system weaknesses. It supports a fine grained analysis of socio-technical failures and considers crucial failure modes. Note that Concerto-FLA also supports modelling of nominal system behavior by assigning “noFailure” to the port in transformation rules. Even if the current implementation of the Concerto-FLA does not implement all the proposed concepts in this thesis, a system architect is able, based on the analysis results, to identify the faulty entities and introduce counter measures if needed.

Although SERA itself addresses the dynamic relations between entities, the current implementation of Concerto-FLA addresses linear relations between the socio-technical entities.

As shown in the case study, the modelling process of Concerto-FLA relies partly on the safety engineers’ experience and their ability to identify socio entities and their failure behavior. However, Concerto-FLA is adaptable to many domains by specifying possible connections between socio-technical entities specified in incidence matrix of organizational and human failures.
8. Conclusion and future work

This chapter presents the concluding remarks of the thesis with highlights of the limitations and possible improvements as future work. More specifically, this chapter is organized as follows. Section 8.1 presents the summary of the thesis while Section 8.2 presents the current limitations and sketched future work.

8.1. Summary

This thesis have introduced a novel model-based approach, named Concerto-FLA, which allows users to perform failure logic analysis on socio-technical systems. Concerto-FLA is built on top of CHESS-FLA and SERA. It supports architects and safety engineers in modelling and analyzing the failure propagation within systems constituted of not only hardware and software components but also organizational and human components. Concerto-FLA succeeded in providing a support for joint analysis of all socio-technical entities. Traditionally, safety engineers had to perform a risk assessment separately for each involved entity. By focusing only on one entity, safety engineers do not have a complete system picture in their mind and could easily miss out on the dependencies between entities. The novelty of the thesis work is reflected in the combination of decomposition of socio entities in terms of SERA classification and representation of their failure behavior in terms of CHESS-FLA FPTC failure modes. This allows the application of existing failure propagation calculus and computing system level failure behavior based on the failure behavior of the internal components. The proposed approach is then incorporated into the CONCERTO tool-set. Moreover, a case study from the petroleum domain is used to demonstrate and discuss the use of the approach.

In this thesis, several approaches have been considered in order to identify the entities that should be modelled in socio-technical systems. Socio entities are first identified as high-level entities inside a socio-technical system. Then, the identified socio entities are further decomposed considering the selected SERA approach. The research works for modelling failure behavior are selected based on the defined criteria whose purpose was to select a suitable approach that allows identification of all socio entities, their relations and defining failure behavior of socio entities.

The identified SERA socio entities that represent failure behavior are then expressed in terms of CHESS-FLA and the underlying FPTC failure modes. This enabled a failure propagation calculus to be performed on technical entities and on newly introduced socio entities. Additionally, to enable an automated analysis, the existing CONCERTO component model had to be adapted to work with new component types.
8.2. Limitations and future work

Besides all the advantages that Concerto-FLA offers, there are several limitations that this approach carries:

- The interpretation of failures of socio entities in terms of CHESS-FLA FPTC failure modes and mapping between organizational and human failures has been proposed with a limited expertise and limited time-frame for this thesis;
- In order to be able to model a system, an analysis should be performed to identify the involved entities and the possible failure types that can occur. Therefore, analysis results are much dependent on the ability of the architect to recognize the possible risks;
- Current implementation of Concerto-FLA allows capturing linear relations between entities which is limiting architects in modelling systems that include dynamic relations between entities.
- Current CONCERTO-ML allows human, organization and technology stereotype to be applied only on blocks, but not on types

The identified limitations revealed a possible direction of improvement that can be taken into consideration. Potential future work and suggestions:

- Therefore, the proposed solution could be possibly reconsidered by domain experts and adapted to the applicable domain. For each applicable domain an incidence matrix should be filled to determine the relations between possible failure types of socio-technical entities. It has to be filled by domain experts and results might be incorporated into the CONCERTO framework as guidelines during the modelling of the system;
- Concerto-FLA should support the modelling of the dynamics of the system and its environment. Therefore, the existing component model should be extended and allow specification and calculation of dynamic aspects in STS. A possible direction could be to adopt the concept of control and feedback loops;
- The usage of the proposed solution has been demonstrated on a simple example from the petroleum domain. A case study should be further developed to further evaluate the accuracy of the proposed matrices in Section 6.3;
- The current implementation of Concerto-FLA does not distinguish entity type in the generated XML files. A possible improvement might be to include the awareness of entity type in XML files and as well in analysis algorithm. An improvement in this direction might support different analysis logic for different entity types;
- Concerto-ML should support assigning human, organization and technology stereotypes on part types as well.
References


44. N. G. Leveson, System safety engineering: Back to the future, Massachusetts Institute of Technology, USA, 2002.


53. P. D. Leedy, and J. E. Ormrod, “Practical research. Planning and design,” ed. 8, New Jersey, Pearson Education Inc, 2005
Appendix A: SERA - Linking active failures with pre-conditions

<table>
<thead>
<tr>
<th>FAILURES</th>
<th>PERSONNEL</th>
<th>TASK</th>
<th>WORKING CONDITIONS</th>
<th>C2 AND SUPERVISION</th>
<th>ORGANIZATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sensory</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Knowledge -</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>perception</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Perception</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Attention</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Communication</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Time management</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Intent - violation</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Intent - non</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>violation</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Knowledge -</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>decision</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Response</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Action</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Selection</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Feedback</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Slips, misses,</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>lapses</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>