Service-Oriented Architectures for Safety-Critical Systems

Abdulaziz Abdulrahman Al-Humam

PhD

University of York
Computer Science

September 2015

Abstract

Many organisations in the safety-critical domain are service-oriented, fundamentally centred on critical services provided by systems and operators. Increasingly, these services rely on software-intensive systems, e.g. medical health informatics and air traffic control, for improving the different aspects of industrial practice, e.g. enhancing efficiency through automation and safety through smart alarm systems. However, many services are categorised as high risk and as such it is vital to analyse the ways in which the software-based systems can contribute to unintentional harm and potentially compromise safety. This thesis defines an approach to modelling and analysing Service-Oriented Architectures (SOAs) used in the safety-critical domain, with emphasis on identifying and classifying potential hazardous behaviour. The approach also provides a systematic and reusable basis for defining how the safety case for these SOAs can be developed in a modular manner. The approach is tool-supported and is evaluated through two case studies, from the healthcare and oil and gas domains, and industrial review.

List of Contents

List of Figures

List of Tables

Acknowledgements

I would like to thank my supervisor Ibrahim Habli for his help, guidance and support throughout my study.

Thanks must go to king Faisal University in Saudi Arabia for sponsoring me during this research.

I would also like to thank my colleagues at University of York particularly: Katrina Attwood, Tim Kelly, Dimitris Kolovos and Richard Hawkins for all their assistance. For their friendship and support, I thank Bestoon Hussien and Nasser Al Malki.

Finally, I would like to express my heartfelt gratitude to my parents and family, in particular my brother Ibrahim Al-Humam, for their constant support during my studies in the UK. I also thank my wife and sons Faisal and Nawaf.

Author's Declaration

Some material presented in this thesis has previously been published in the following paper:

Ibrahim Habli, Abdulaziz Al-Humam, Tim Kelly, Leila Fahel: Integrating Safety Assessment into the Design of Healthcare Service-Oriented Architectures, Medical Cyber Physical Systems Workshop, Berlin, Germany, April 2014.

All of the work contained within this thesis represents the original contribution of the author.

This work has not previously been presented for an award at this, or any other, University. All sources are acknowledged as References.

Chapter 1

1. Introduction

1.1. Service-Oriented Architectures (SOAs)

Many organisations are primarily focussed on providing services to their customers. Notable service-based domains include banking, retail, healthcare and transport. A service can be described as a function which is well defined, is self-contained, and often does not have any internal dependency on the context or status of any other functions (1). A service has been defined as a "*value delivered to another through a well-defined interface and available to a community*" (2).

One approach to designing and representing service-based systems, especially for software-based services, is through Service-Oriented Architectures (SOAs). SoAs provide *"a paradigm for defining how people, organizations, and systems provide and use services to achieve results*" (2). SOA was introduced as a design paradigm to minimise the gap between the business process requirements and the IT services implementing and supporting these requirements. SOAs are commonly designed with flexibility, dynamism and the capability to respond to changes within a system's operational environment (3).

SOA-based systems are typically constructed using patterns, which provide reusable template solutions for common problems encountered within a given domain. Patterns offer 'road maps' to build scalable applications, by breaking a problem down into smaller solvable problems (4). Patterns are generally

refined and optimised, depending on the specific context of the system being designed, to provide a possible solution for the problem being investigated (5).

One of the formal definitions of SOA was provided by the Organization for the Advancement of Structured Information Standards (OASIS) group and the Open Group as follows:

> "[An SOA is] *a paradigm for organizing and utilizing distributed capabilities that may be under the control of different ownership domains. It provides a uniform means to offer, discover, interact with and use capabilities to produce desired effects consistent with measurable preconditions and expectations*." (6) (7).

SOAs have been lauded as a means to reduce the complexity required in the infrastructure of a solution platform by transforming complex business processes and complicated IT systems to a set of building blocks, i.e. services (8).

Although there are different interpretations of SOA terminology and concepts for different perspectives, most agree on a set of common characteristics. These characteristics include loose coupling, service contracts, autonomy, reusability, composability and discoverability. These are discussed in more detail in Chapter 2.

SOA is increasingly being used in the development of enterprise systems (3). From the business perspective, SOA aims to facilitate the design of applications that are able to provide better services without increasing resource consumption, since the service provider can provide the same service to multiple clients concurrently. Consequently, more efficient systems with improved response times are expected (9).

Within an SOA setting, service providers should have standardised communication methodologies, which allow the various business services provided by different vendors to communicate smoothly with each other. This allows businesses to interact more efficiently, potentially reducing the cost and time required to perform tasks as compared with traditional applications (10).

For some in the software engineering community, SOA is a way of thinking which can help to prompt the design of dependable systems that address flexibility, dynamicity and Quality of Service (QoS) as metrics to analyse business processes (4). Among the important characteristics of SOA are loose coupling and on-demand integration, which potentially allow organisations to become more proactive and dynamic. However, this brings challenges concerning the management and assurance of QoS, e.g. security, safety and performance.

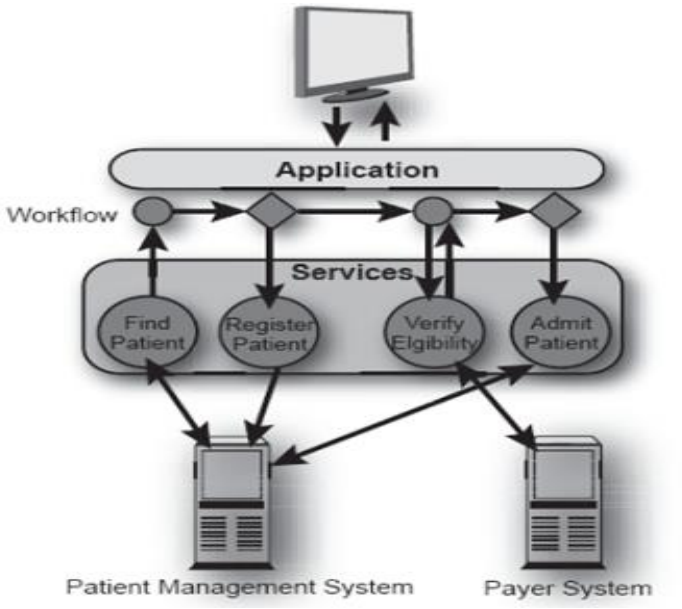


Figure 1: A Conceptual View of Healthcare Services (11)

Figure 1 shows a simplified example of an SOA-based system from the healthcare domain (11). Two systems, Patient Management System and Payer System, communicate and provide basic services within a private healthcare setting, which are as follows:

- Register patient;
- Admit patient;
- Verify Eligibility;
- Find patient.

The above services support key care processes within a modern health organisation. However, services can fail to achieve their intended behaviour

(e.g. providing incorrect patient information). When used in a safety-critical setting, service failures can directly lead to hazards and potentially harm.

1.2. SOA and Safety

Safety is one of the most significant dependability qualities for modern systems, since the consequences of hazardous failures can include harm to people, damage to property or threats to the environment (12). The assurance of safety requires specialised concepts, standards, methods and tools. Service-based systems used in safety-critical applications have to satisfy their safety requirements at each level of service to support clients and other composite services and ensure that the safety risk posed by the system is acceptable. They also have to comply with any applicable safety regulations or assurance standards.

Safety risks can be categorised as resulting from failures by humans involved in or interacting with the system, or failures due to system components and interactions (12). Leveson introduces absolute safety as "*freedom from accidents or losses*". However, since this is not reasonable in the real world, Leveson recommends the adoption of Lawrence's definition of a "safe" system as one in which the risks involved are deemed acceptable (13).

Understanding and controlling the complex relations between system behaviour and the emergence of safety risks is a significant challenge. Addressing this challenge at the structural and behavioural level requires collaboration between the different technical and organisational stakeholders, e.g. users, analysts and system engineers. Insufficient interaction between domain experts and engineers remains a key difficulty in achieving effective risk assessment.

An important aspect of the safety engineering process is providing assurance that the risk of hazardous behaviours of the system (and its components), which are identified in the risk assessment process, have been identified and managed appropriately to ensure safe system behaviour. This assurance is commonly communicated in the form of a safety or assurance case (14). Safety cases provide a reasoned and structured argument to demonstrate why the

available evidence, generated from testing and analysis, supports the claims made about system safety. In a complex environment, the overall safety case might be compositional (15), integrating separate safety argument modules for the various services, functions and sub-systems. In the context of such a system, although services might in reality be provided by sub-systems developed by different organisations, they are clearly interdependent and so are their corresponding safety arguments (16).

1.3. The Challenges of SOA in Safety-Critical Systems

SOA is increasingly being adopted in high-criticality domains (16). In sectors such as healthcare and aviation, many services are categorised as safety-critical, and present various types of hazards and failures. An SOA designated for use in a safety-critical application must be shown to satisfy its safety requirements and developers and assessors must ensure that the safety risk posed by the system is acceptable before it can enter service. However, there is a lack of systematic techniques for generating safety requirements for SOA from the application of safety analysis. There is also a lack of systematic approaches for providing assurance that those requirements have been met for the SOA. This can hamper the adoption of SOA as an approach to building large-scale safety-critical systems.

The research presented in this thesis aims to address safety analysis and safety cases for safety-critical SOA, focusing on both the individual services and their associated interactions in order to identify hazardous behaviours associated with the system. Specifically, the following challenges are addressed in this research:

- There is a shortage of research on the use of SOAs for the development of safety-critical systems;
- There is a lack of systematic safety assessment processes and methods that address the special features of SOAs, such as composibility;
- There is a lack of safety assurance strategies to guide the development of safety argumentation and evidence to justify why the use of SOAs in safety-critical applications is acceptably safe.

1.4. Research Hypothesis

Following from the challenges described in the previous section, the hypothesis proposed in this thesis is as follows:

> ***Integrated*** *architectural modelling and analysis of safety-critical service-oriented systems provides a* ***traceable****,* ***consistent*** *and* ***systematic*** *means for* ***assuring*** *the safety of these systems.*

The following explains the key terms used in the thesis hypothesis:

- **Integrated:** safety analysis and design decisions evolve together;
- **Traceable:** forward and backward model-based traceability between design and safety analysis is provided;
- **Consistent:** design features are reflected in the safety analysis and the safety assurance and vice versa;
- **Systematic:** a methodical approach to performing safety assurance is proposed;
- **Assuring:** the approach provides an explicit means for reasoning about the safety properties and system behaviour.

The above terms define the basis for the criteria used for evaluation in Chapter 6. This hypothesis is supported by the following four contributions made as part of this PhD research:

- Identification and configuration of **SOA modelling notations** based on two criteria, namely coverage of the structure and behaviour of the services that are deemed suitable for the safety analysis and assurance of SOA.
- Development of **safety analysis methods for SOA** which can be used to examine the hazardous failures of services and the failure behaviour of lower-level implementations.
- Development of the concept of safety cases for the assurance of safety-critical SOAs, supporting modularity and promoting correspondence between the structure of the SOA and the organisation of the safety case.

This is defined in a **SOA Safety Argument Pattern Catalogue**, which can be instantiated to produce safety arguments for specific SOA-based systems.

- Implementation of **tool-support** that provides automated capabilities for building the SOA models, applying the safety analysis and building the safety case for the system.

The target audience of the approach comprises software and systems engineers responsible for designing the architecture and for modelling the design. The safety analysis and assurance activities will more likely be the focus of the safety engineers. Of course this cannot be achieved without engaging the domain experts (e.g. clinicians in healthcare or chemical engineers in oil and gas). Therefore, the collaboration of domain experts and systems, software and safety engineers is important for performing the analysis and assurance activities in our proposed framework. In particular, the framework targets early lifecycle activities, mainly requirements engineering and architectural design (both in terms of structural and behavioural architecture) and their integration with the safety activities, e.g. hazard analysis.

1.5. Thesis Structure

The thesis is divided into the following chapters:

Chapter 2 presents a survey of the published literature in relevant areas of the SOA paradigm. The survey first looks at the core elements of SOA. This includes a review of typical concept definitions, features and lifecycles of SOA and covers SOA design patterns and modelling. Next, the chapter reviews safety and certification. This includes an examination of the common elements of the safety engineering process, such as safety analysis, risk assessment, risk management, safety requirements and safety tactics. Next, the review considers safety assurance and safety standards, with particular attention to the concept of safety cases, including argument representation and modularity. Chapter 2 concludes with a review of literature concerning dependability and safety of SOA and other related systems.

Chapter 3 introduces the conceptual basis for this research by presenting our proposed SOA Safety Assurance Framework. The Framework defines how SOA

modelling, safety analysis, safety cases and tool support are performed in an integrated way. The chapter introduces the modelling languages that we use to specify the services and their underlying processes. It also introduces the methods we developed and adapted for identifying service hazards and potential failure behaviour in an SOA system design. The chapter proposes the use of modular safety cases and patterns to define the different argument structures for assuring services and their design and for constraining variation in SOA safety cases. Finally, the chapter presents an overview of the automated support developed and the tooling environment used in this research for the modelling, safety analysis and assurance of SOA.

Chapter 4 explains the SOA safety analysis methods adapted and developed in this thesis. These methods consider potential service hazards and failure behaviours of services and tasks according to the ways in which these services and tasks are defined in the SOA context. This in turn forms the basis for the definition of safety requirements which can help influence or drive the design processes. A healthcare case study is introduced to provide an illustration of the application of the SOA safety analysis methods we have developed, and a preliminary evaluation of their effectiveness. The last section describes the implementation of the tool support that provides the capabilities for modelling, safety analysis and traceability.

Chapter 5 develops a catalogue of safety argument patterns for justifying the use of SOA in safety-critical applications. The overall safety argument pattern is created in a modular manner to ensure a clear correspondence between the structure of the safety reasoning and the organisation of the SOA design. These argument patterns link the safety argument for the SOA with the results of the modelling and safety analysis presented in Chapter 4. Finally, we illustrate the application of the safety argument patterns to the services defined within the healthcare case study and provide a preliminary evaluation of their effectiveness.

Chapter 6 describes our evaluation methodology in detail. It also presents a second case study, in which the SOA Safety Assurance Framework is applied to an industrial system in the oil and gas domain. A safety argument is produced,

based upon the SOA modelling and safety analysis described in Chapter 4 and the safety argument patterns presented in Chapter 5. Finally, Chapter 6 contains a critical evaluation of the results of this PhD research.

Chapter 7 presents the main conclusions of this research and discusses potential areas of future research.

Chapter 2

2. Survey of SOA and Safety

2.1. Introduction

This chapter provides a survey of the published literature relating to the topic of this thesis. The survey is split into three main sections. The first section reviews current literature relating to the basic concepts of SOA, as well as to architectural design and the notations commonly used in developing SOA-based systems. The second section discusses the basic concepts relating to system safety engineering. This includes the methods and analysis techniques available to system developers. Finally, we explore the concept safety cases and review literature relating to the development of safety arguments.

2.2. SOA

2.2.1. SOA Definition

Generally, in the context of software, a service can be defined as:

"*One application or computer program performing some work for another application or computer program. This work may include some functionality or data sharing*" (17).

The definition is richer in the context of SOA, where services are defined as well-defined and self-contained functions. When services are combined to form sets, there are policies to control their usage and relationships to each other and whether dependability attributes are required (1). Generally, each service in an architecture represents a high-level business concept (18).

The concept of SOA has been considered from several different perspectives (19) (3) (4). For example, SOA can be regarded as a means for developing

systems based on flexible concepts, which are sufficient to guarantee the operability of services within the system to meet business demands (3). However, Josuttis stated that, regardless of the dissimilarities in defining SOA found across the literature, the introduction of improved flexibility is an advantage of SOA which is highlighted in all definitions (4). Moreover, Josuttis pointed out that SOA is only a paradigm; when it is applied, specific decisions should be taken, which differ according to the situation being investigated.

2.2.2. SOA Entities

Typically, an SOA system comprises six entities, namely: consumer, registry, provider, contract, lease and proxy (20) (21). These entities interact with each according to a predesigned paradigm to ensure adequate provision of defined services. Figure 2 shows the find-bind-execute paradigm (20). Generally, the find-bind-execute paradigm allows the consumer of a service to request a third-party registry for the service which is compliant with its standard. When the registry has acknowledged a service, it provides the consumer with a contract specifying an endpoint address to the service. SOA has six entities that are configured jointly to support the find-bind-execute paradigm, as we explain in the following subsections (20).

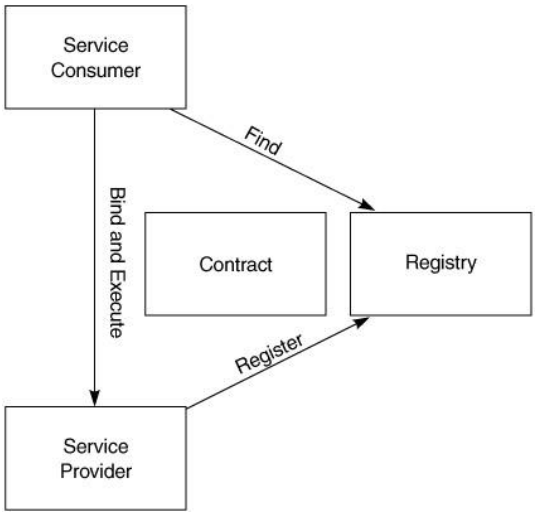


Figure 2: Find-Bind-Execute Paradigm (20)

Service Consumer: A service consumer is any element that requires a service, e.g. a software element, an application or other services. The service consumer should have the required functionality to find the registry, bind to the service

provider and execute the required function(s). Communication with the provider and the executed function(s) must comply with the service contract.

Service Provider: The service provider element is responsible for accepting and executing the consumer requests. Prior to performing any functionality within the system, the service provider should register its contracts with the service registry.

Service Registry: The service registry is a directory that is located on the network, and holds all contracts from the service providers and makes them available for the service consumers.

Service Contract: The service contract defines the specification of the service, which includes the specified formats for the consumer request and the provider response, the QoS level, and the pre- and post-conditions of the contract.

Service Proxy: The service proxy is typically a set of APIs, located on the service consumer, and is responsible for finding the contract and the service provider on the registry, and for performing the necessary communication with the provider on behalf of the consumer.

Service Lease: The service lease defines the validity period of the contract, starting at the time at which the registry offers the contract to the consumer.

2.2.3. SOA Characteristics

Most approaches to SOA design agree on a set of common characteristics. These characteristics are as follows.

Loose Coupling: According to SOA, a system can be built using a composition of different related but independent services. This will permit the services to be managed individually (3). Consequently this will allow loosely-coupled services, which will permit the development of flexible and dynamic applications, which are formed by composing these services according to predefined requirements (4).

Service Contract: Service providers will publish the contracts for the available services on the service registry. Service consumers will use these contracts to

bind and execute the services. SOA adheres to contract design principles, to ensure standardization of the service-providing mechanisms. The standardization will enable consistency of view for the service (3) (22).

Autonomy: Although services are self-contained and independent and have full control over the execution logic that they contain, when services are combined to form compositions it is important that they are able to perform the required tasks or processes. Thus, information exchanged between services, using messages, should fulfil the necessary underlying requirements that enable these compositions to perform the designed tasks (3).

Reusability: Business processes are typically decomposed into tasks, each of which will be performed inside the system by a service. It is possible that a single task will appear in different processes. Reusability is the concept of developing a flexible service that can be integrated into different compositions or business processes. This will allow for higher productivity of the development team and minimise the time required to produce the implementation required for these services, as the reusable service need only be implemented and tested once (23).

Composability: By building services with enough capabilities to interact effectively with the environment, compositions can be formed by integrating the different services to provide a higher level service. This will allow faster building of SOA applications, as composed services will inherit some functions and attributes of the services inside the composition (24).

Discoverability: One of the main principles of the SOA design paradigm is to introduce applications that can interact and respond to changes in the environment. One of the main characteristics of SOA applications is their ability to introduce within the system services that can be integrated and published in the registry automatically, and discovered by the service consumer (24). Therefore, new services, replicas of current services, and new versions of current services can be introduced into the system according to the service consumer's demands.

The above characteristics make SOA a challenge for use in safety-critical applications. More specifically, each service can be treated as a self-contained element in a larger system (i.e. autonomy) and therefore it is important to identify the hazards that it can pose (as an individual service). At the same time, a service is part of larger system and therefore we need to understand the nature and the safety impact of interactions between the different services and the wider environment (i.e. contracts and loose coupling). These characteristics are shared with conceptual paradigms such as Systems of Systems (25). However, SOAs can be treated as a more specialised design framework, with its own design features and structures, e.g. Service-Level Agreements and Participants. Furthermore, unlike object-oriented technologies (26) or product lines (27), which tend to be used at a more detailed design and implementation levels, each service in an SOA-based system tend to be visible at the user-interaction level and can be changed during the runtime phases on the SOA-based system.

2.2.4. SOA Lifecycle

Traditionally, software is developed in well-defined stages, at each stage, a certain set of tasks should be performed to help ensure that the final software meets the business objectives (28). The development of software based on the SOA paradigm is different, as it includes both the business and the IT vision within the software. Thus it is reasonable to add extra levels of governance to control the execution of the loosely coupled services (29). According to Cox and Kreger, the SOA lifecycle is gradually evolving to a form in which more integrated processes are clearer (30).

Figure 3 shows the SOA solution lifecycle according to Cox and Kerger (30). The figure shows that the lifecycle is divided into two main stages, the preproduction and production stages. The preproduction stage includes the model definition and implementation; the production stage includes mapping the model to the target infrastructure, monitoring the results and reacting to issues which arise. Seeley also cites Manes, Research Director at Burton Group Inc., who added that an initial stage including service identification and commitment is also required (29).

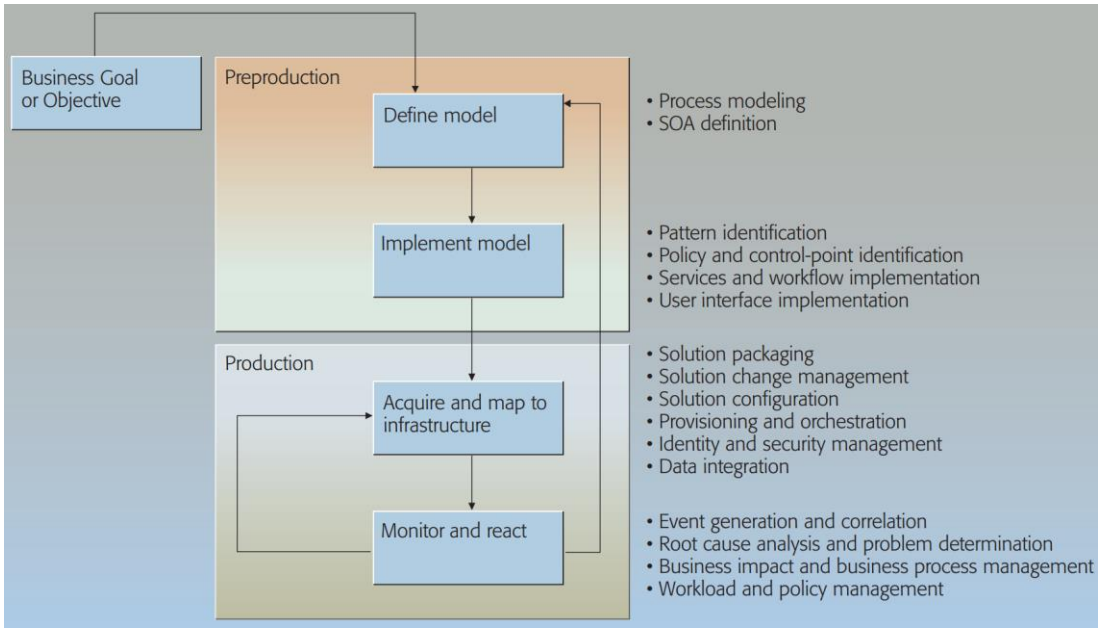


Figure 3: SOA Solution Lifecycle (30)

Kumar (31) defined the different life cycles of the SOA application from the Enterprise implementation perspective, defining the service lifecycle as part of the enterprise SOA implementation lifecycle. Kumar asserts that the enterprise SOA implementation lifecycle includes three stages: initiation, development of the roadmap and execution plan. At each of these stages there are different tasks to be performed, these tasks relate to the different lifecycle stages defined using other perspectives.

A management framework for the SOA lifecycle was introduced by Gejnevall (32) (shown in Figure 4). The framework divides the lifecycle into six broad areas in each of which four distinct phases are identified. The six areas are architecture, processes, information, solution, services and infrastructure. The four phases are preparation, development, and delivery and governance. The framework identifies the different tasks that should be performed during each phase, for each of the 6 lifecycle areas.

An alternative way of representing the SOA design lifecycle is provided by Ramollari (33). Here five lifecycle phases are identified: analysis and design, construction, testing, deployment and governance. According to Ramollari, some or all of these stages will be included in any SOA development.

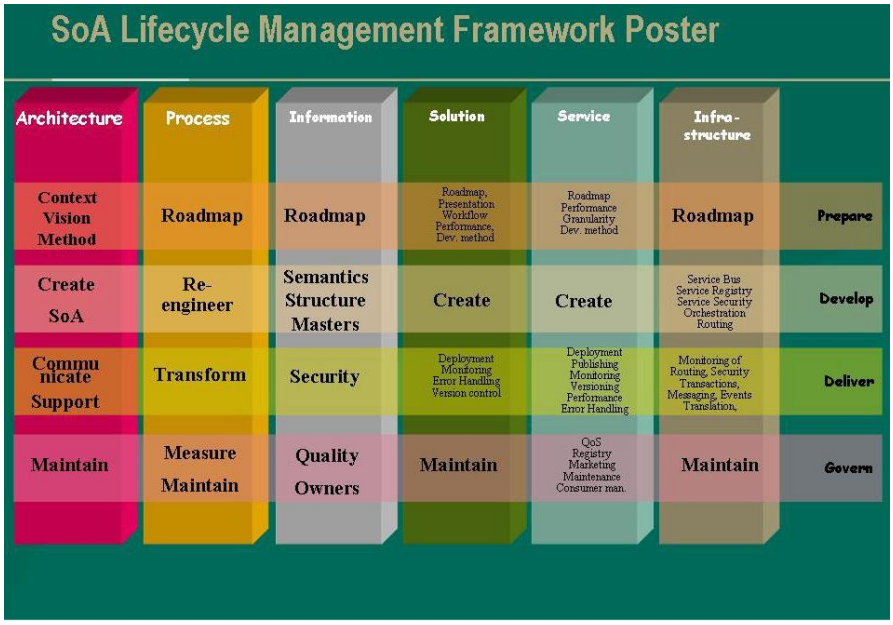


Figure 4: SOA Lifecycle Management Framework (32)

Analysis and Design:

At this stage, the business requirements will be analysed. The required business functionalities will be mapped to the underlying IT functions or services. Also at this stage, the business objectives will be addressed by the SOA design strategies. Part of this stage is defining the timelines and deliverables of the SOA. The last activity at this stage is performing the modelling to the designed architecture.

Construction

The input for this stage is the specification of the designed architecture. The specification will be used to develop the implementation of the services and the necessary environment in which to operate these services. This includes the necessary protocols to allow the different services within the application to interact and the required interfaces to allow other applications to connect to the application and to perform their permitted services.

Testing

It is important to review the different functionalities implemented within the application. Testing the functionalities and services which have been developed is essential to ensure the provision of a reliable application that meets the business requirements.

The diversity of the hardware infrastructure, operating systems and applications that comprise the environment of the SOA application will increase the complexity of the testing process (34). Moreover, this complexity will increase because SOA software is flexible and can change according to the target environment. Thus, it is important to ensure that the testing process can correctly address the characteristics of the SOA software being tested and produce accurate results that are readable and informative.

Deployment

Once the suitability of the developed application has been ensured, it is logical to deploy the application within the targeted environment. This includes both operating the SOA application as a standalone application and connecting the application with other applications that are operating in the environment, to test for correct behaviour and interactions.

Governance

Different activities should be investigated at this stage. Most important is the development of a governance model which will allow control of the different components of the SOA software. As well as this, it is important to identify the business and IT transformations.

2.2.5. SOA Design Patterns

2.2.5.1. Definition

In software engineering, design patterns are often used to provide, and exploit, reusable solutions for common problems encountered in software design. A design pattern is a description or template to solve a general problem, which can be used in different circumstances (35). The original idea of the design pattern was created by an architect, Christopher Alexander, who named and described common architectural and landscape planning problems and discussed their solutions (36). Alexander asserts that:

> "*Each pattern describes a problem which occurs over and over again in our environment, and then describes the core of the solution to that problem, in such a way that you can use this*

solution a million times over, without ever doing it the same way twice." (37).

Drawing inspiration from Alexander's work, (37) developed a pattern language for software engineering. They suggested that communication between the components of the environment should be achieved in order to undertake a customised solution for the problem being investigated (37). Software design patterns therefore identify and define the basics of a design structure, which is common or context-specific. These design structures will be represented as objects, and these objects will be reused to compose new design objects.

While Alexander indicated that the pattern should describe the core of the solution, design of the real solution for any problem is should be based on different factors, e.g. the current circumstances of the environment. This is particularly important for design patterns specified for the software domain. Further, Erl (5) in his definition of the design pattern mentioned that the solution provided for any problem should be proven and individually documented.

2.2.5.2. Use of Design Patterns

Design patterns are very important for building high-quality applications, since they allow the designer to reuse the different elements of the software, instead of re-inventing the wheel for each element. Moreover, reusability of elements should lead to reductions in the time required to introduce the software, as it will enable the developers to use tested and certified solutions which have been applied before.

2.2.5.3. Design Pattern Elements

For any pattern there are four essential elements: name, problem, solution and consequences (38) (37).

Pattern Name: Unique naming of any element in the software domain will make it easier to identify the pattern. It can be identified clearly in any discussion, without confusion (38).

The Problem: This describes the problem and details of the circumstances of the environment in which it occurs (37).

The Solution: The different elements, which when combined introduce the design, are called the solution. Documentation of the solution requires well-defined and clear descriptions for the elements, their responsibilities and all of the relations and collaborations required between them.

The consequences: It is also important to document the trade-offs introduced in applying the pattern, which are used to evaluate the different design patterns, and select the most effective solution to the problem being investigated (37).

2.2.5.4. Design Pattern Categories

Design patterns can be classified into categories based on the perspective adopted by the author in proposing the pattern. Metsker and Wake (39) defined five different categories of design patterns based on their primary intent: interface, responsibility, construction, operations and extensions.

Table 1 gives examples of standard software engineering patterns in each of these categories.

Intent	Patterns
Interfaces	Adapter, Facade, Composite, Bridge
Responsibility	Singleton, Observer, Mediator, Proxy, Chain of Responsibility, Flyweight
Construction	Builder, Factory Method, Abstract Factory, Prototype, Memento
Operations	Template Method, States, Strategy, Command, Interpreter
Extensions	Decorator, Iterator, Visitor

Table 1: Sample Categorisation of Design Patterns by Intent (39)

Other authors categorise the design patterns according to the application areas to which they can be applied. Most commonly, these areas are defined as: creational, structural and behavioural patterns (40) (37).

2.2.5.5. Design Patterns and SOA

Design patterns existed within the software design domain before the emergence of the SOA paradigm. For example, there are object-oriented, Enterprise Application Architecture and Enterprise Integration Applications (EIA) design patterns. Design patterns that fall within the SOA pattern catalogue can be classified into new design patterns, existing design patterns or

design patterns that are intentionally similar to patterns in other catalogues (5).

SOA design patterns have been defined in the following areas: service, service composition, service inventory, and service-orientation enterprise architectures (5). In structuring a taxonomy of SOA design patterns, each of these four groupings can be further divided into sub-parts.

Table 2 shows an outline taxonomy for SOA design patterns, with examples of the available design patterns for each group. The fourth group, service-orientation enterprise architecture is a composition of the other three architectures.

Architecture	**Design patterns**
Service	
Foundational service patterns	Service -Identification Patterns Functional Decomposition, Service Encapsulation Service-Definition Patterns Agnostic Context, Non- Agnostic Context, Agnostic Capability
Service implementation patterns	Service Facade, Redundant Implementation, Service Data Replication, Partial State Deferral, Partial Validation, UI Mediator,
Service security patterns	Exception Shielding, Message Screening, Trusted Subsystem, Service Perimeter Guard,
Service contract design patterns	Decoupled Contract, Contract Centralization, Contract De-normalization, Concurrent Contracts, Validation Abstraction,
Legacy encapsulation	Legacy Wrapper, Multi-Channel Endpoint, File Gateway
Service governance patterns	Compatible Change, Version Identification, Termination Notification, Service Refactoring, Service Decomposition, Proxy Capability, Decomposed Capability, Distributed Capability
Service composition	
Capability composition patterns	Capability Composition, Capability Recomposition
Service messaging patterns	Service Messaging, Messaging Metadata, Service Agent, Intermediate Routing, State Messaging, Service Callback, Service Instance Routing, Asynchronous Queuing, Reliable Messaging, Event-Driven Messaging
Composition implementation patterns	Agnostic Sub-Controller, Composition Autonomy, Atomic Service Transaction, Compensating Service Transaction

Service interaction security patterns	Data Confidentiality, Data Origin Authentication, Direct Authentication, Brokered Authentication
Transformation patterns	Data Model Transformation, Data Format Transformation, Protocol Bridging
Service inventory	
Foundational inventory patterns	Inventory Boundary Patterns: Enterprise Inventory, Domain Inventory, Inventory Structure Patterns: Service Normalization, Logic Centralization, Service Layers, Inventory Standardization Patterns: Canonical Protocol, Canonical Schema
Logical inventory layer patterns	Utility Abstraction, Entity Abstraction, Process Abstraction.
Inventory centralization patterns	Process Centralization, Schema Centralization, Policy Centralization, Rules Centralization
Inventory implementation patterns	Dual Protocols, Canonical Resources, State Repository, Stateful Services, Service Grid, Inventory Endpoint, Cross-Domain Utility Layer,
Inventory governance patterns	Canonical Expression, Metadata Centralization, Canonical Versioning

Table 2: Outline Taxonomy of SOA Design Patterns (5)

2.2.5.6. SOA Design Pattern Considerations

As already emphasised, SOA is a design paradigm for computer-based applications, with defined goals and objectives for applications, which meet the corresponding goals and objectives identified from the business perspective. Design patterns should be selected, designed, refined and applied to allow SOA-based applications to meet their goals and objectives.

The different elements that comprise the SOA application should be able to interact with each other without requiring any intermediate transformation of exchanged contents. Intermediate transformations would increase the application's complexity and the time required to perform any operation.

From the business perspective, Return-On-Investment (ROI) is a major factor in selecting any solution or product. By developing services that are able to support each other, which can be composed according to the requirements and

designed to meet the inventory architecture, an organisation can increase the ROI for each of the services it provides or exploits.

There are other factors from either a business or IT perspective that are important for the selection of a design pattern from the available alternatives, e.g. vendor diversification, business and technology alignment and IT costs. The ability of the IT solution to meet the business requirements will be directly affected by these factors, so it is very important to prioritise them and develop the SOA application based on that prioritisation (5).

2.2.5.7. Dependability and SOA Design Patterns

This thesis is particularly concerned with three attributes of dependability are which need to be considered developing any SOA application for a safety-critical domain: availability, safety, and security. Various design patterns have been developed to address issues related to these attributes within the different architectures of the SOA. Also, these dependability attributes can be considered to as part of the general superset of Quality of Service (QoS) attributes. Before we discuss dependability design patterns, we explore the concept of dependability, and how it related to SOA, in the next subsection.

2.2.5.8. Dependability

Different factors that allow the development of more dependable computer systems have been identified, for example availability, reliability, and maintainability (41). More interest in the dependability of computer systems was shown during the 1980s, as a result of increasing integration of these systems into the business arena. The concept of dependability builds on several related concepts, in particular reliability and fault-tolerance. Dependability can be defined as "*the trustworthiness of a computing system which allows reliance to be justifiably placed on the service it delivers*" (42).

As more IT technologies were introduced, two fundamental attributes of dependable systems were added to the ones which had already been highlighted: safety and security (43).

There are different perspectives on the attributes that inform dependability. The total list of attributes comprises integrity, reliability, safety, security,

confidentiality, maintainability and availability. According to the point of view adopted in the context of a particular system or domain, some of these attributes may be eliminated or combined (44).

Avizienis, et al. (45) defined threats as the intentional or accidental events that affect the provision of a correct service. Service failure, within a computer system, is defined as the deviation of the delivered service from the correct service due to the occurrence of an event. This may result in the external systems interacting with the service to enter incorrect or unknown states, which would be observed as anthe error. The presence of the error is due to the presence of a fault in the system (45). It is therefore possible to define different states that are based on the presence of error or failure in the service, as an indication that the service is not performing correctly. For example, degraded, slow, limited and emergency service modes can be defined to reflect the severity of the failure within the system. Figure 5 shows one of the mostly widely used dependability trees.

To attain and maintain the different attributes of the dependability, different means have been developed. The means can be classified into four main groups: fault prevention, tolerance, removal and forecasting. Fault prevention and fault tolerance allow the system to provide a trusted service. More confidence in this service will be achieved if it can also provide fault removal and forecasting. Dependability attributes are targeted by tactics (46) that are used to achieve a certain level for each attribute. Requirements for the attributes will be derived, according to the architectural decisions resulting from the application of the tactics.

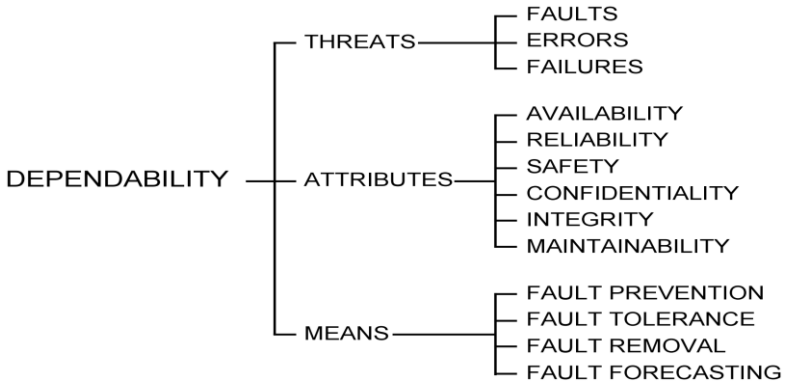


Figure 5: Dependability Tree (45)

Failures in dependable systems often result in losses. These losses can be related to customers, sales, revenue and productivity. Moreover, there will be an additional cost incurred in bringing the system back into normal operation (47). When software is one of the system components, it is possible to have failures within the system caused by the presence of the software. In such cases, failures might be limited to the software component of the system, or might cause the system as a whole to fail. Thus it is important to address safety as part of every stage of the software lifecycle. As an example, in 2011, a software failure in a Pak 'n Save supermarket in New Zealand resulted in the store's doors opening automatically at a time where there were no employees inside, allowing the public to leave with free groceries (48).

Security Related Design Patterns

Security is of great concern when exposing a local service to outside users. External users can use services to hack the internal network or to manipulate the resources managed by these services. The **Service perimeter guard** design pattern addresses this issue by introducing an intermediate service at the network perimeter, which can be used as an interface between the internal service from one side and external users from the other side. In this case, the intermediate service will be a secure contact point for the local network; thus, the security threats related to this issue can be minimised (5).

Other design patterns have been defined to address other security issues and threats. These patterns fall into two main groups. The first group, service security patterns, addresses concerns in the service architecture. This group includes the **exception-shielding, message-screening, trusted subsystem and service perimeter guard** patterns. The second group, service interaction security patterns, addresses concerns in the service-composition architecture. Patterns in this group include **data confidentiality, data origin authentication, direct authentication and brokered authentication**. For more information please refer to the discussion of SOA design patterns in Erl (5).

Availability Related Design Patterns

Some SOA design patterns provide solutions for the availability attribute. These solutions can be selected according to the requirements of the service. For example, in the group of service-implementation patterns, there is a **redundant implementation** pattern which addresses the case of a heavily-used service. In this case, the service can be considered as a potential source of failure in the system, especially if it is being used in composition with other services. Any fault the service faces will be implicated in all compositions that it is part of. Based on this, the redundant implementation pattern provides the use of redundant implementation or failover support as a solution to this problem.

For more information about other SOA design patterns providing solutions for the availability attribute please, refer to the discussion of SOA design patterns catalogue introduced in Erl (5).

Safety Related Design Patterns

In order to ensure that a system's safety requirements are satisfied appropriately, it is important to ensure that the runtime behaviour of all components within the system complies with their specifications at the design and development stages. To help satisfy this, **monitoring design patterns** are introduced. Monitoring design patterns help in building tools that are able to monitor the components within the system and to measure to what extent the specifications used at system development are respected at runtime (49).

Although safety is a key concern in ensuring the appropriate operation of the different software applications within a safety-critical system, there has not been a great deal of work on safety design patterns for SOA (50). Different techniques have been developed to provide a certain level of system safety, which are not specific to SOA. For example, **partitioning** allows the isolation of different independent components of the software, to ensure that severe faults that may occur in any component are contained. Generally, safety tactics are designed and organized based on the fault-handling mechanism. There are three handling mechanisms, failure avoidance, detection and containment (50).

For example, the **Triple Modular Redundancy (TMR) pattern** provides a solution to avoid a single point of failure for a processing unit, by implementing three processing units based on the same input, along with a voter that decides which unit's results will be transferred to the output (51). The voter will detect failures occurring in any of the processing units and the final result will be defined according to the results from the three processing units.

Different variations of the TMR pattern have been implemented to ensure the avoidance of failure, for example, by implementing three independent inputs for each processing unit and three voters, each of which takes input from the three processing units. Logically, this will improve the system's integrity in the face of failure, as it reduces the likelihood of system failure due to a single failure or a failure of any of its components.

Jackson (52) argued that current practices in developing software indirectly address dependability attributes in general and safety in particular. This appears to be true, since design patterns that provide specific solutions for safety are very rare. Thus, as Jackson suggested, new tools or practices should be developed to address safety-related issues directly. Kang and Jackson discuss an approach to dependability analysis based on trust bases, which offers a promising way to address dependability assurance more directly (53).

2.2.5.9. Service Level Agreements

A Service Level Agreement (SLA) is a document that can be used to define the relationship between two parties: the provider (or producer) and the recipient (or consumer) (54). This is an important item of documentation for both parties. Its purpose is as follows:

- Identify and define the customer's needs;
- Provide a framework for understanding;
- Simplify complex issues;
- Reduce areas of conflict;
- Encourage dialog in the event of disputes;
- Eliminate unrealistic expectations.

An SLA is a core part of a service contract, in which the specification and quality of service are formally defined. An important role of an SLA is to capture the non-functional aspects of the service so as to monitor the performance of the service to ensure that it is used according to the contract between different parties. According to (55), SLAs are used for the preservation of service capacity. Increasingly, SLAs are used as a way to ensure delivery of services in IT-based SOAs. During SLA discussions, all parties must agree on the interface and the service provided by the provider. If the provider is providing a service that was not originally assured or agreed upon, then we have a SLA disagreement fault (56). One way of representing SLA is using a tabular format as shown in Table 3 (57).

Between IT Department And ABC Department				**Date: MM/YY**	
Contacts: **IT Department:** _____ **ABC Department:** _____				**Effective Dates: From MM/YY to MM/YY**	
Approvals: **IT Department:** _____ **ABC Department:** _____					
CICS Service			**Goal**	**Actual**	**Difference**
	Availability	8 am - 5 pm Mon-Fri	98%	100%	2%
	Response	% of response within 2 seconds (internal)	90%	95%	5%
		% of response within 2 seconds (internal)	95%	95%	0%
	Load	Transactions/min during peak (9 am – 11 am)	300	250	-50
		Daily CPU hours	3.5	3.0	-.5
	Accuracy	Errors due to DC Problems	0	0	0
		Errors due to Applications	0	0	0
	Batch Service	Class S: % turnaround in 30 minutes	95%	85%	-10%
		Class T: % turnaround in 15 minutes	98%	100%	2%

Table 3: SLA Sample (57)

2.2.6. SOA Modelling

After decisions regarding the architectural design have been reached and agreed on, it is necessary to provide an unambiguous record of the decisions. This is often carried out in the form of modelling (58). Modelling can be defined as a description of some phenomenon of interest (59). In the case of SOA, the

phenomenon is the service to be provided. Kolovos asserted that models can be classified into unstructured and structured:

> "*Unstructured models are descriptions that do not adhere to well-defined syntactic or semantic rules. Examples of this type of model are board drawings, informal sketches, natural language specifications etc. Due to their lack of conformance to well-defined rules, unstructured models are only useful for communication between humans. By contrast, a structured model is an artefact that conforms to a set of syntactic and semantic well-formedness constraints which is called its modelling language or metamodel. Due to their rigorously defined nature, except for human communication purposes, structured models are also useful for mechanised processing*". (59).

The models used to describe a system are based on system characteristics such size, complexity and heterogeneity (58).

The system architecture model is the composition of the different architectures that represent the system's constituent elements and structures. Viewpoints are system architecture models that are constructed according to the perspective of the stakeholder. Viewpoints can be constructed using Architectural Description Languages (ADLs) (60).

Software modelling can help developers better understand the system and allows for effective and efficient organisation. Based on this understanding, different design alternatives can be explored and considered based on their importance.

Different ADLs are introduced to enable designers to build different versions/aspects of the model and to help the development team to define optimized code. Each modelling language represents a different point of view on the system. Examples of these languages include the Unified Modelling Language (UML) (61), the Systems Modelling Language (SysML) (62), the Specification and Description Language (SDL), and the Business Process Modelling Notation (BPMN) (63).

Different modelling languages are introduced to ensure proper modelling of SOA solutions. Service-Oriented Modelling and Architecture (SOMA), Service-Oriented Modelling Framework (SOMF) and Service-oriented Architecture Modelling Language (SoaML) are examples of these languages.

SOMA was introduced by IBM as the first modelling language for SOA (64). SOMA aids in the analysis and design of SOAs through the identification, specification and realization of the services that are required to build the system. SOMA also provides the ability to compose services from preliminary services.

SOMF was introduced to provide modelling of SOAs; it is based on conceptions and a universal language (65). Strategic business decisions can be used along with IT tactics to design and provide solutions to enterprise problems. According to the developer of SOMF, it can be used to provide modelling for any application and include any business processes or technological services.

UML was introduced as a general modelling language, thus no capabilities were introduced for SOA modelling. The OMG introduced SoaML, as a UML profile and metadata that adds the required capabilities to allow precise modelling of SOA based on UML.

In the next sections, we discuss two of the most commonly used modelling languages and comment on their applicability to SOA.

2.2.6.1. BPMN

The BPMN standard (currently at version 2.0) was developed by the Business Process Management Initiative (BPMI) (66). In May 2004, version 1.0 of the specification of BPMN was released to the public and in February, 2006, BPMN 1.0 was adopted as an OMG standard. BPMN is a graphical modelling notation for business processes in an organisation. White (66) discussed the context and use of BPMN, considering the four main elements of BPMN: flow objects, connecting objects, swim lanes, and artefacts. Figure 6 illustrates the graphical elements of BPMN. Flow objects are displayed as a set of elements that include different types of events, activities, and gateways. An event means “something that occurs” as distinct from an activity which is defined as “something that is

done" (67). Gateways can be used to control the communication between the flows of paths via splits and joints.

Connectors can be used to model the flow between objects. The connector consists of sequence flows, message flows and associations objects. Sequence flows can be used for ordering the activities that are performed in a process; message flows can be used for sending and receiving messages across businesses; and association can be used for linking information with flow objects. Swim lanes can help in classifying and managing activities. There are two types of swim lanes: pools and lanes. Artefacts can be used to add more information to the model. Artefacts comprise data objects, text annotations, and groups. The data object can be used to manage the input and output data for activities, the text annotations can be used to describe the process, while groups can be used for grouping certain activities (67).

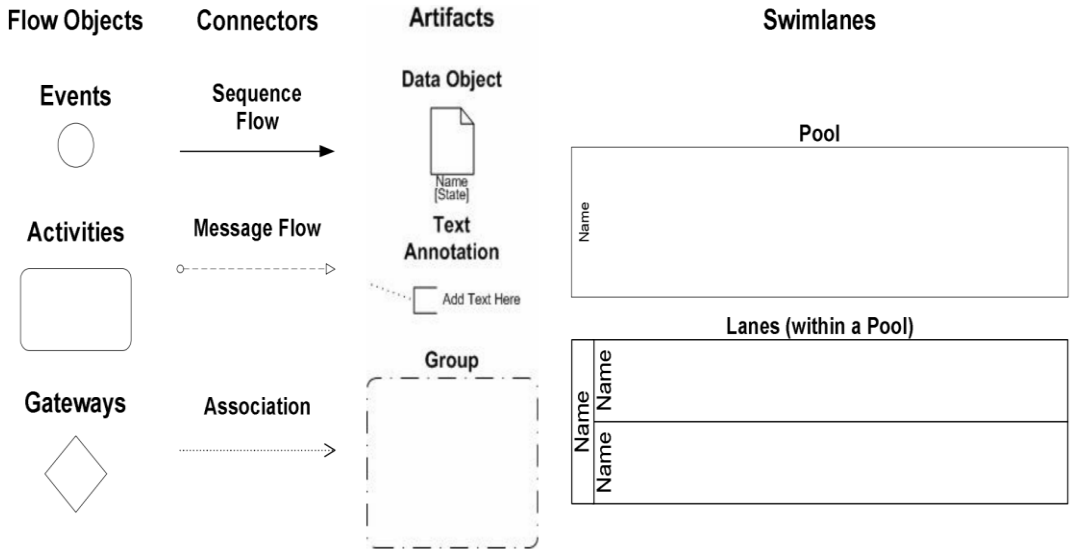


Figure 6: Basic BPMN Elements (66)

2.2.6.2. SoaML

The OMG introduced SoaML as a UML profile which has the capability to model SOA (68). The SoaML profile is structured around five SoaML diagrams, namely, the Service Participant diagram, the Service Contract diagram, the ServicesArchitecture diagram, the Service Interface diagram and the Service Categorisation diagram. Each of these diagrams provides a different view to define and understand services (68). For example, the ServicesArchitecture diagram shown in Figure 7 allows for the definition of Participants and

Services, showing the collaboration between different organisations, and the roles each Participant plays within each ServiceContract (collaborationUse).

The Service Interface diagram shows the different types of Interface which the SoaML standard offers. Figure 8 shows a simple interface, which describes a one-way service which does not require protocol, e.g. a service can be used without knowing anything about the entity which calls it.

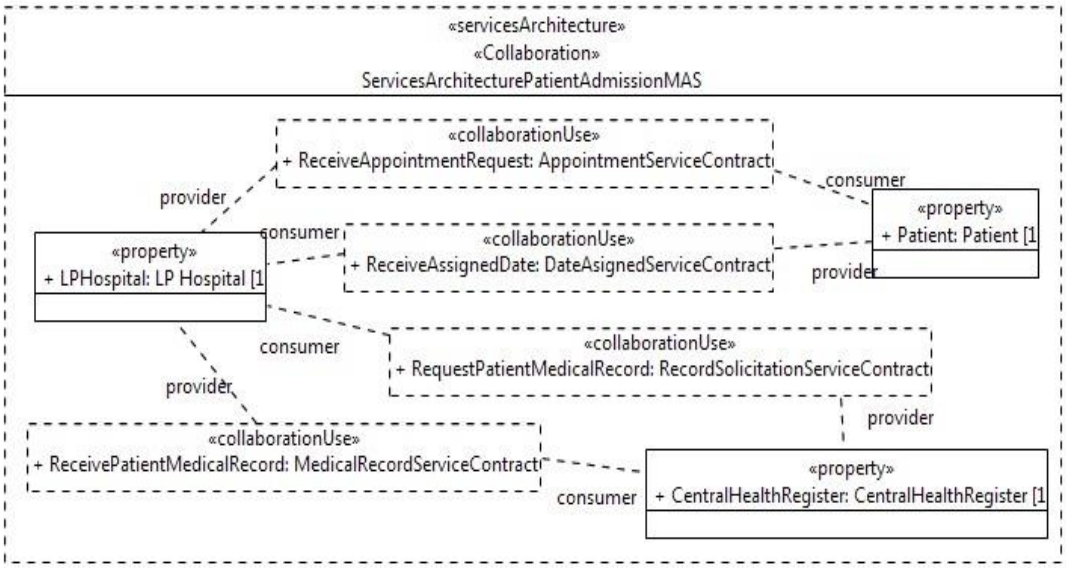


Figure 7: ServicesArchitecture Diagram in SoaML

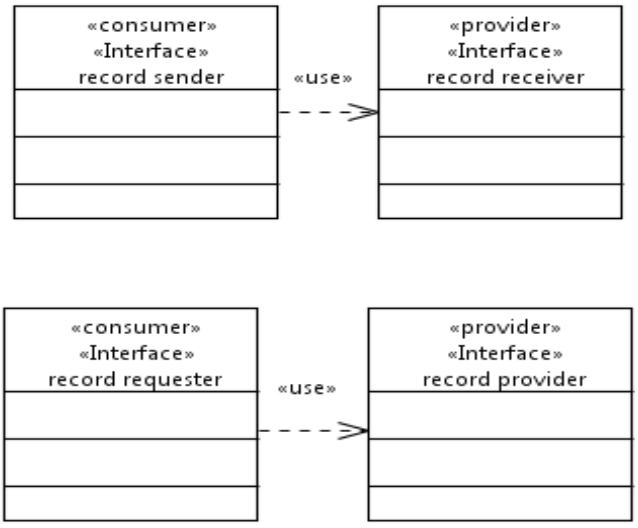


Figure 8: Interfaces Diagram in SoaML: UML Simple Unidirectional Interface

As Figure 9 indicates, a service interface can also be defined for bi-directional services. Here, the service includes communication between the consumer and

the provider of the service which may be required to complete the services. Further information about UML simple Interface unidirectional, UML interface bidirectional, and ServiceInterface, which is defined by understanding and using UML simple interfaces, is provided in (69).

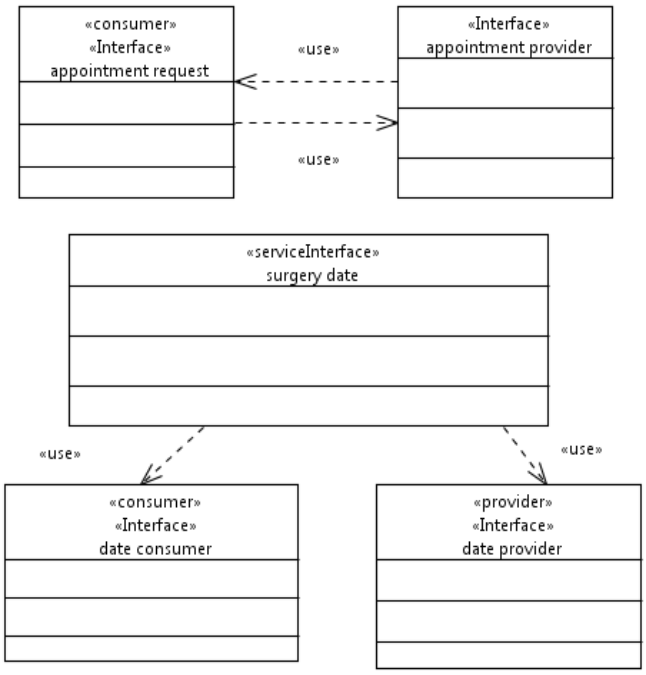


Figure 9: shows a service interface and a bi-directional service interface

2.3. Safety and Certification

This section introduces the concepts of safety, safety-related issues, and the certification processes. Section 2.3.1 provides a review of safety in safety-critical systems. A review of aspects relating to the certification of such systems will be presented in Section 2.3.2

2.3.1. System Safety

Roland, et al. (70) defined system safety as applying special technical and managerial methodologies to guarantee systematic and forward-looking control of hazards during the lifecycle of the system being investigated. Roland et al. suggested that safety analysis and hazard control activities should be performed during the conceptual phase of the system, with further safety actions defined for the lifecycle stages.

The concept of safety within the system is related to the physical context of the system. A system can be considered unsafe if its effects are harmful for humans. Logically, software components can cause harm to humans indirectly. Hazards or accidents within the system occur because of three main sources: humans who interact with the system, the system itself or the environment in which the system operates. For example, an accident between two cars on a road could be caused by human error (e.g. excessive speed), a failure in one of the cars (e.g. sudden failure of the lights at night) or causes within the environment (e.g. heavy fog).

In order to manage the definition and implementation of safety requirements for a system, it is important to understand which of the safety activities should be integrated within each stage of the lifecycle. Figure 10 shows a "V" system development lifecycle, along with the safety activities related to each stage (71). The figure shows the different stages of the development lifecycle: requirements analysis and specification, architectural design, detailed design, implementation, integration, testing and verification, and delivery and commissioning. At each stage, the associated safety activities are integrated in the development process. For example, hazard identification and risk assessment, which is a continuous activity, is commonly performed as part of requirements analysis and common cause/common mode and zonal analyses are associated with the detailed design and implementation stages.

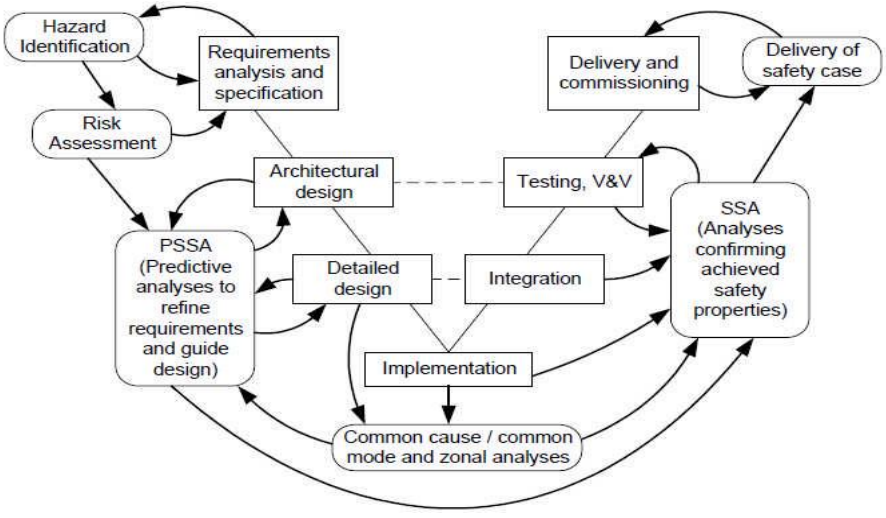


Figure 10: A "V" System Development Lifecycle (72)

2.3.1.1. Safety Processes

For safety-critical systems, it is important to perform an analysis of the hazards, identify them, evaluate their effects and provide the necessary measures to manage the hazards. Pumfrey defined a hazard as a "physical situation, often following some initiating event that can lead to an accident" (72). The hazard analysis determines the extent to which each hazard can lead to an unsafe system status.

Safety analysis helps to define the necessary requirements that ensure the system being developed is acceptably safe. Safety analysis should be carried out as part of a safety management process, which will result in examining the system safety level and whether the system can be deployed or not.

The safety process should be designed to address relevant factors within the system and its environment, for example, the size of the company, the structure of the system, the industry the system is operating in and the applications the system is used for (12). Each stage should result in a detailed documentation of the system safety process results.

2.3.1.2. Safety Analysis

A preliminary hazard list should be provided for safety-critical systems. Based on the hazard analysis, a list of high-level safety requirements, specifying how the hazards should be managed and controlled in design and operation of the system, is typically generated. These requirements are then transformed into detailed safety criteria. The system safety team should decompose these detailed safety criteria into readable detailed requirements which can be used by the design team to develop the system (73). Safety analysis should be performed throughout the system lifecycle to identify the hazards and risks posed by the system, assess each of these hazards or risks, and provide and recommend solutions for managing the risks.

The safety analysis should be based on the milestones defined for the safety processes and should comply with the milestones of the system development process. The reporting of the analysis should address any challenges faced in

performing the analysis, such as lack of information or insufficient recommendations.

Safety analysis techniques can be classified into two main categories, deductive and inductive. Deductive analysis helps in answering the question, “How can something occur?” Conversely, inductive analysis helps in understanding the consequences of an event, which has either occurred or has been postulated. Thus it helps in answering the question, “What if this happens?” (74). Some techniques can be used either inductively or deductively, depending on the context. Markov Analysis (MA) and Fault Propagation and Transformation Calculus (FPTC) are examples of these types of technique (27).

Generally, failures in the system can be classified into either random or systematic failures. Typically random failures occur in the system due to faults in the hardware components. Such faults can be caused by different issues, such as the age of a component. Systematic failures, on the other hand, are the repeatable under the same system and environmental conditions, as a result of a human error that could occur at any stage of the system lifecycle. However, a fault due to an error of this kind will not occur unless the system is in an appropriate state (75).

Due to their nature, random faults are often measured quantitatively, while system failures are measured qualitatively.

Failure Modes and Effects Analysis (FMEA) is one of the main inductive safety analysis techniques. Pumfrey (72) discussed the categorisation of the FMEA analysis technique and techniques that have been developed using it as a basis. Simply, starting from a certain event (which has either occurred or is postulated) FMEA performs an analysis for the different consequences that might be faced.

FMEA provides a basis for understanding how the requirements defined in the preliminary stages of the development are implemented in the system, and of whether the system actually meets these requirements.

To perform FMEA, a set of information needs to be collected. This information can be identified and categorised according to the issue being addressed.

Information collection should be organised to ensure that the information needed to satisfy the analysis requirements is assembled. Different forms have been designed for data collection, such as the example shown in Table 3 (76). Stamatis states that the information within the form can be categorised into three parts. The first part is introductory and is essential to the analysis process; however, none of the information in this part is mandatory. The analysis is completely dependent on the information in part 2, thus all of the items in this section are mandatory. More items can be added to the form or the items in the form can be reorganized according to the analysis requirements, but none of the items in the form can be removed. The third part contains the signatures of the team performing the analysis. Although it is not mandatory, it reflects the team's responsibility. Stamatis numbers the parts according to the analysis requirements performed. These numbers can be altered as required.

(1) Process name: _____ (3) Involvement of others: _____ (6A) Key prod. date: _____
(1A) Part name: _____ (4) Supplier involvement: _____ (7) Prepared by: _____
(2) Mfg or design responsibility: _____ (5) Model/product: _____ (8) FMEA date: _____
(2A) Person responsibility: _____ (6) Engineering release date: _____ (9) FMEA rev. date: _____
Page____of____pages

Process function	Potential failure mode	Potential effect(s) of failure	∇	S E V	Potential cause(s) of failure	O C C	Detection method	D E T	R P N	Recommended action	Individual/ area responsible and completion date	Action results				
												Action taken	S E V	O C C	D E T	R P N
(10)	(11)	(12)	(13)	(14)	(15)	(16)	(17)	(18)	(19)	(20)	(21)	(22)	(23)			
Team effort																

(24) Approval signatures _____ _____ (25) Concurring signatures _____ _____

Table 4: FMEA Form (76)

2.3.1.3. Key Safety Activities

There are key issues that need to be investigated when analysing the safety of a system. Firstly, it is important to understand possible hazards which might lead to the presence of risks within the system. These risks should be assessed and any necessary steps taken to manage them. In the following three subsections, hazard analysis, risk assessment and risk management will be discussed.

Hazard Analysis

Pumfrey asserts that hazard identification, which implies locating the potential hazards within the system in order to specify suitable management and control measures, is the first step in the safety process.

Different systems pose different hazards. Some of these hazards can be considered as generic, or common for all systems, whether computer-based or not. Examples include environmental hazards, like floods, fog, storms and earthquakes. However, possible failure of altitude measuring sensors is a hazard specific to aircraft for example. However, whether the hazard is generic or specific, the context of the system dictates the way in which it is handled. For example, fog is a hazard for both aircraft and cars; however, in each system the solutions will be designed differently.

The process of hazard analysis includes several steps that are designed to address the goals of the analysis (12). Hazard analysis can be performed on planned or existing systems. However, for most systems, complete elimination of hazards is not feasible and therefore hazard control measures are used for those hazards which cannot be eliminated.

Different techniques have been developed to perform hazard analysis. These techniques use either quantitative or qualitative methods to evaluate the hazard, based on hazard characteristics like hazard level (the degree to which the hazard will affect the system) and likelihood.

Functional Failure Analysis (FFA), Hazard and Operability Analysis (HAZOP) and Fault Hazard Analysis (FHA) are among the best known hazard analysis techniques (12) (72). There are also other techniques, and variations of these techniques which are designed to identify and analyse hazards. However, Pumfrey argued that these techniques are not widely accepted in industry and are limited to use within the research community, so we do not consider them in detail here.

Functional Failure Analysis

Functional Failure Analysis (FFA) is an inductive technique which is intended to derive the hazards from a functional description of the system (sometimes referred to as Functional Hazard Analysis) (77). For each function of a system,

FFA examines all possible failure modes of the function that may contribute to unsafe behaviour and provides suitable recommendations for mitigation in the design (72). The FFA technique consists of four steps as described by Pumfrey:

- Identifying a function;
- Identifying the failure modes of the function, based on three potential deviations: function not provided, functional provided when not required and incorrect function;
- Assessing the effects of each failure mode at various abstract levels;
- Suggesting requirements and recommendations to improve the design.

Table 4 shows an example of hazard analysis of a vehicle speed sensor using FFA. The results of FFA are commonly presented in a tabular format.

Function	**Failure mode** a) not provided b) provided when not required c) provided incorrectly	**Effects**	**Comments and Recommendations**
Vehicle speed sensor	a) No vehicle speed data provided to engine management system	Vehicle speed used in alternative load calculations and crank speed check routines. Effect minimal, and restricted to loss of some error detection capability.	No significant safety effect.
	a) No vehicle speed data provided to gearbox controller	Vehicle speed is primary input to gear selection, including kick-down. Effect is significant reduction in accuracy and timeliness of gear shifts. May impair driveability of vehicle, especially in start-stop traffic.	Potentially hazardous. Driver warning essential; loss of speed display on dashboard is not sufficient. Design must incorporate detection (via comparison with engine speed / load data?) and illumination of transmission warning light.
	b) No meaningful interpretation	N/A	Both controllers sample VS data when needed, therefore data cannot be provided when not required.
	c) Inaccurate vehicle speed data provided to engine management system	Effect minimal, and restricted to loss of some error detection capability.	No significant safety effect.
	c) Inaccurate vehicle speed data provided to electronically-controlled automatic gearbox management system	Effect is unpredictable. May result in unwanted gear shifts or selection of inappropriate gear. May impair driveability of vehicle, especially in start-stop traffic.	Potentially hazardous. Depending on degree of inaccuracy, failure mode may be hard / impossible to detect. Investigate acceptability of failure rates of specified components.

Table 5: Example of Hazard Analysis of a Vehicle Speed Sensor Using FFA (72)

HAZard and OPerability Studies (HAZOP)

Hazard and Operability Studies (HAZOP) is a predictive safety analysis technique that was developed in the mid-1960s, to assist, assess and refine the

design of new process plants (78). HAZOP was originally developed at ICI, for use in the chemical industry (79). The main focus of this method is on the flow of chemical material and liquids between different components of a chemical plant, Table 6 shows the steps of the HAZOP process. The process provides a set of guide words to help the analyst to consider a variety of possible deviations for each flow. Although the original HAZOP guide words were designed for use in analysis of processes within chemical processing plants, they are sufficiently generic to be applicable to other safety-related systems, with some interpretation.

Step	**Description**
1	Select a flow in a pipeline.
2	Identify important physical attributes of the flow, such as pressure, temperature, flow rate, chemical composition etc.
3	Consider the deviations prompted by applying each guide word to each property for this line section.
4	Determine the possible causes of each of these deviations.
5	Investigate the expected outcome (effect on the plant) of each deviation, taking into consideration operating conditions and other causal factors where necessary, and examining the contribution of protection mechanisms and other mitigation already included in the design.
6	Decide which deviations are safety problems (i.e. those with both plausible causes and hazardous effects).
7	For deviations which are not safety problems, record a justification (i.e. explain why the design is acceptable as proposed.).
8	Consider changes to the plant that will remove, or reduce the probability or severity of, hazardous deviations.
9	Determine whether the cost of the proposed changes is justified.
10	Agree actions and responsibilities.
11	Repeat steps 1 to 10 for all other lines in the plant.
12	Follow up to ensure necessary actions have been taken.

Table 6: Steps of HAZOP (72)

The standard guide words for process plant analysis (with example interpretations) are shown in Table 7.

Guide Word	Deviation	Example Interpretation
NO or NONE	No part of the intention is achieved	No forward flow when there should be.
MORE	Quantitative increase in a physical property (rate or total quantity)	Higher pressure, flow rate, temperature... Quantity of material is too large.
LESS	Quantitative decrease in a physical property (rate or total quantity)	Lower pressure, flow rate, temperature... Quantity of material is too small.
MORE THAN or AS WELL AS	All intentions achieved, but with additional effects (qualitative increase)	Impurities in flow (air, water, oil...) Chemicals present in more than one phase (vapour, solid)
PART OF	Only some of the intention is achieved (qualitative decrease)	One or more components of mixture missing, or ratio of components is incorrect
OTHER THAN	A result other than the intention is achieved	Any state other than normal operation, e.g. startup, shutdown, maintenance...
REVERSE	The exact opposite of the intention is achieved	Reverse flow.

Table 7: HAZOP Guide Words (72)

HAZOP differs from the other safety techniques we have reviewed in that it is a team-based activity. A HAZOP team typically comprises around six people who are selected for different roles and interests, generally including the following:

- The plant design engineers;
- Commissioning or installation engineers;
- Operators and maintenance engineers;
- One or more independent experts;
- HAZOP leader;
- Recorder or secretary.

In its original form, HAZOP does not work well for systems where the flow involves information. This fact has led to the development of Software Hazard Analysis and Resolution in Design (SHARD) for the analysis of computer-based applications (72).

Software Hazard Analysis and Resolution in Design (SHARD)

Pumfrey (72) proposed a technique called Software Hazard Analysis and Resolution in Design (SHARD), which is a refinement of HAZOP intended to address the requirements of safety analysis in computer systems (hardware and software).

Table 8 shows the steps of the SHARD process. The process provides a set of guide words to help the analyst to consider a variety of possible deviations for each information flow.

Step	Description
1	Identify and consistently label the flows in the diagram. For each flow, identify its source (where the flow originates) and sink (destination). Assemble any additional material from data dictionaries, timing diagrams etc. required to completely describe the operation of the flow.
2	Review the design to ensure that the intended operation is clear.
3	Construct a table of guide words, interpreting the six major failure classifications for the combinations of communications protocol and data type used in the design.
4	Work through the flows in a systematic order, using the combination of protocol and data type to select a set of guide words from the table constructed in step 3, and considering the deviations from intended behaviour suggested by each guide word.
5	Determine and record the potential causes of the suggested deviations. Note that identifying the causes of a deviation will require study of the active component (process) or data store that is the source of the flow
6	Determine and record the expected effects of the suggested deviations. Except at the top (context) level of a design, the extent of identification of effects should be limited to the active component (process) or data store which is the flow sink. Effects that propagate beyond the immediate destination of the flow will be considered as part of the analysis of further data paths that originate from this component.
7	Reduce the set of suggested deviations to a set of meaningful failure modes by discarding those for which the potential causes are acceptably improbable, and those for which no hazardous effects have been identified. Note that whenever a deviation is discarded, a justification must be recorded
8	For each meaningful failure mode identified, suggest alternative management strategies. These may take the form of design modifications to remove its causes or limit its effects, or a set of requirements that must be satisfied by lower-level design elaboration to achieve acceptable system-level safety properties.
9	Select of one of these strategies to pursue, and record a justification for the selection. In the worst case, if no acceptable management strategy can be suggested, the only acceptable course of action may be a redesign.

Table 8: SHARD Process

SHARD can be used for the analysis of information flows in a normal operation and can also study the possibility of unintentional behaviour. The guide words defined for SHARD (72) are as follows:

Omission:

There is no delivery of the service; no communication (service not provided).

Commission:

A service is provided when it is not required, the communication is unforeseen (more services provided).

Early:

The service (communication) is provided earlier than required

Late:

The service (communication) is provided later than required

Value:

The service or data has the wrong value. Errors of this kind can be divided into two classes: detectable (the problem can be detected and protected or mitigated against) or undetectable (the problem cannot be found).

SHARD does not need a team to carry out each step of the analysis. The results can be captured in a tabular format. The columns should include "Guide word, Deviation, Possible causes, Detection and protection, Effects, and Justification or design recommendations". Table 9 shows a sample of SHARD outputs for a computer-assisted braking system (72).

Risk Assessment

Once the hazards have been identified and recorded, the potential risks within the system can be identified and analysed. Upon identifying the risks, it is important to perform risk assessment and management. Risk assessment includes different tasks. The first is a detailed interpretation of the identified risks, such that risks can be classified into acceptable and unacceptable risks. Second, reduction measures for the unacceptable risks should be proposed. Typically, these measures should be selected to control the hazards involved.

During the risk assessment process, the system design will be revised, to ensure that risks are reduced. For risks where it is not possible to eliminate the hazards by changing the design, the measures will be proposed ensure that these risks will be minimized to an acceptable level. Risk assessment techniques can be quantitative or qualitative, depending on the requirements of the assessment.

For example, the presence of fog will increase the risk of car accidents. Fog limits visibility, making it difficult for drivers to the road or other vehicles clearly. For a car at a speed of 40 miles per hour with visibility of 100 metres, the risk of having an accident due to fog is limited, or can be classified as acceptable. But for the same car with a visibility of 10 metres, the risk of having an accident is high or unacceptable.

Flow ID 1 **Source** OUT **Data Type** Record (4 x pressure values)

Flow Name Actuator_control **Destination** Hardware **Protocol** Signal

Additional Information Written by OUT processes on all 3 processors. First to complete is used by hardware, rest are ignored.

ID	Guide Word	Deviation	Possible Causes	Co-effectors	Detection / Protection	Effects	Haz Cat	Justification/Design Recommendations
1.1	Omission	No output records sent. Results in no change in braking pressure (stuck at).	Failure of OUT process on all 3 processors, or failure of both CAN buses	N/A	None possible	a,b,d,e	Cat	Rec: Ensure reliability of processor hardware and CAN bus implementation adequate. Must be at least 1 order of magnitude better than hydraulic components of system.
1.2	Commission	Additional output records sent. Result is unwanted change in braking pressure.	Spurious CAN messages.	N/A	Only has effects when spurious message is valid, but with out-of-sequence ID.	b,c,d,f	Cat	Rec: Check integrity of message ID generation in OUT process. CAN protocol with CRC checks etc. considered sufficient to prevent incorrect interpretation of other CAN messages.
1.3	Early	N/A	N/A	N/A	N/A	N/A	N/A	Just: Meaningless. Output can never precede initiating change in demand.
1.4	Late	Output records sent late.	Failure of scheduling software or clock hardware on all 3 processors. CAN bus overloading.	N/A	N/A	b,e	Cat	Rec: Clock failure only considered likely as result of failure of synchronisation procedure. Check integrity of synchronisation. Check possible causes of CAN bus saturation.
1.5	Late	Unacceptable response time to change in demands. (end-to-end latency)	Scheduling failure of tasks / messages	N/A	N/A	b,e	Cat	Rec: Investigate watchdogs to limit process over-run. Ensure there is no way in which BASIC process can be prevented from completing every cycle.
1.6	Value, detectable	N/A	N/A	N/A	N/A	N/A	N/A	Just: Hardware has no error detection / correction capability. All value errors must be considered undetectable.
1.7	Value, undetectable	One or more components of output record not proportional to demand	Undetected algorithmic failure in any process. Corruption of CAN message.	N/A	CAN protocol. TMR on BASIC processes.	a,b,c,d,f,g	Cat	Rec: Apply every possible defensive programming strategy to detect and reject incorrect values.

Table 9: Sample SHARD Output for a Computer Assisted Braking System (72)

Risk Management

The process of defining the hazards, analysing them, and performing assessment on the risks caused by the hazards will result in a better understanding of hazards, risks and potential losses arising from their occurrence. Procedures for avoiding or minimising losses can then be proposed.

Understanding of the risks inherent in a system will allow these risks to be managed. The process of risk management is aimed at mitigating identified risks and achieving the desired goals of the system (80). According to Hillson, the success of risk management can be measured by the effectiveness of the selected solutions in achieving the final goals.

Although effective risk management will increase the chances of avoiding the risks within the systems, resulting in the best achievement of the objectives, even in the presence of risk management, the system will have residual risks. Hillson et al. argued that risk management processes, tools and techniques should evolve over time to be able to address changes in the system and its environment. They also defined three major period of evolution in the attitudes to risk enshrined in standards, before 1997, from 1997 to 2000 and after 2000 (80).

- **Before 1997**: the concentration was on the negative side of the risk; risk is equivalent to threat in this period.
- **From 1997-2000:** the definition of risk starts to add the positives or opportunities along with the negatives.
- **After 2000:** all standards appearing after 2000 includes both threats and opportunities as part of risk management.

Risk management includes the identification, communication and resolution of possible risks within the system (81). It can be divided into two main steps. The first is performing risk identification and assessment. The second is risk control. Risk assessment has been discussed above, so only risk control will be discussed here. Risk control is the process of planning risk resolution techniques, providing solutions for potential risks within the system and finally

monitoring the applied solutions to ensure their effectiveness in resolving the risks. Figure 11 shows the risk management cycle as proposed by Kasse Initiatives (81).

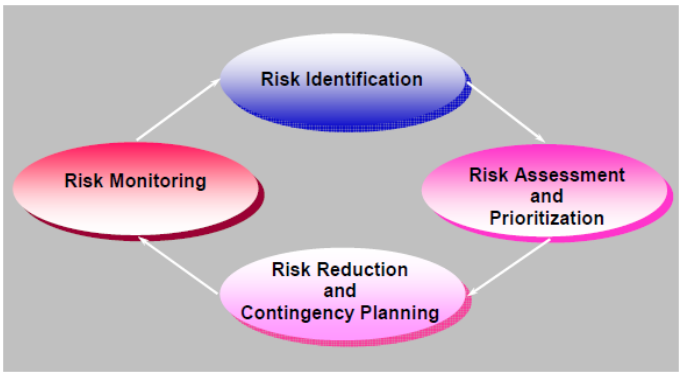


Figure 11: Risk Management Cycle (81)

The processes, tools and techniques used when performing risk management should be selected to suit the system and the risks being addressed. This will ensure that resources are wisely consumed by the risk management process and the risk management solutions will be practical and efficient.

Risk management can be implemented within the system using four different approaches (82). These are:

- **Inactive risk management:** the system is assumed to work normally and nothing will be done regarding the risk.
- **Reactive risk management:** no risk assessment is carried out in advance; risk management will be performed based on the presence of accidents.
- **Interactive risk management:** risks will be considered when they have a high potential to occur, in other words, risks will be considered based on the phase the system is currently in, as risks depend on the current system status.
- **Proactive risk management:** risk identification, assessment and solving will be performed completely in advance of the system development. Thus, solutions for potential risk will be implemented as part of the system design.

As the approaches change from the inactive to the proactive, the time and cost overheads of risk management will increase. However, the possibility of suffering loss due to risk will be minimized by the proactive approach.

2.3.1.4. Safety Requirements

One of the outputs from the risk assessment and management activities is a set of recommended solutions for the expected risks in the system. These solutions will be used to define and develop the safety requirements of the system.

It is important to ensure that the system is acceptably safe, by guaranteeing that the system meets its safety requirements. Within each industry different standards have been developed to help assure the safety of the products. For example in aviation systems, Preliminary System Safety Assessment (PSSA) uses the risk assessment and management to ensure that the system being developed is acceptably safe (72). PSSA is applied at the design phase of the system to guarantee a system that meets the safety requirements and maintains refined safety requirements that are appropriate to the final design of the system (83).

As discussed in the system development lifecycle, hazard identification and analysis and risk assessment activities are performed during the first stage of system development, which is requirements analysis. These activities will define the safety requirements of the system. Risk management is the process of integrating the safety solutions as part of each system development stage.

For example, the safety requirements relating to environmental hazards, like lightning for an aircraft, will be defined during the system requirements analysis. The design and operational solutions provided for these hazards will be integrated in the aircraft system during the different stages of development, for example to avoid crashing due to lightning, the aircraft body should include metal which makes it operate as a conductor, this solution will be integrated as part of the design and implementation stages.

2.3.1.5. Safety Design Tactics

The achievement of high quality and performance (e.g. design decisions) in systems depends on many factors. So, we need a strategy to assist in recording

and examining the system design decisions. Tactics are strategies which help to capture these design decisions. According to Bass et al. “A tactic is a design decision that influences the control of a quality attribute response.” (46).

Safety is an important quality attribute for many computer systems. Different architectural patterns can be used to build each component within the computer system. Safety tactics are the foundational building blocks that define which design patterns among those available can be selected to achieve the attributes’ requirements. Wu (58) asserted that safety tactics can be used to select the architectural design patterns that allow the safety requirements to be met. Wu discussed the use of redundancy, voting and diversity tactics with the **Triple Modular Redundancy** (TMR) pattern to avoid a single point of failure and to increase the availability of the system, which guarantees a higher level safe system. Figure 10 shows a taxonomy of safety tactics according to Kelly’s and Wu’s classification.

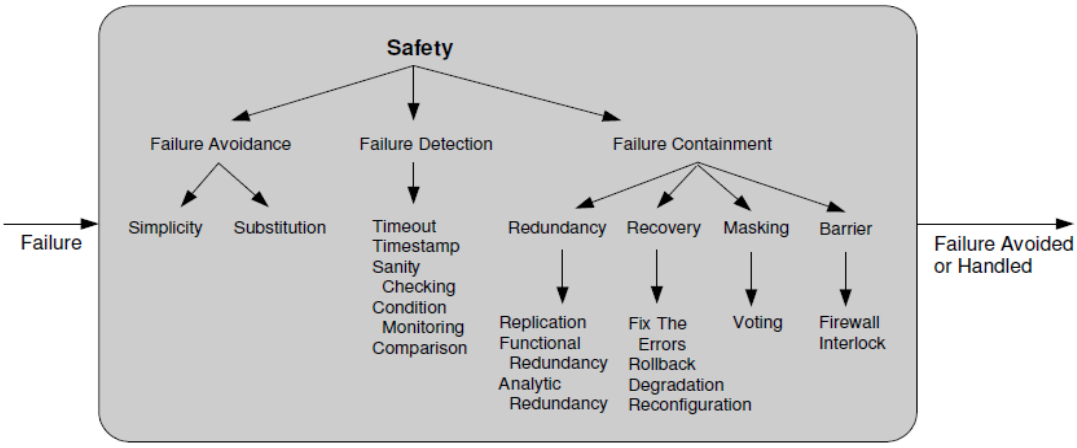


Figure 12: Safety Tactics (50)

Safety is related to reducing the consequences of undesired accidents. It is best to avoid failures and, in cases where failures occur, they should be detected and managed. Safety Tactics can be classified into three groups: failure avoidance, detection and containment tactics (84).

Failure avoidance is the best mechanism to handle failures: if a failure is avoided, it cannot occur. **Simplicity** tactics are designed to help in designing systems that can avoid failure. Logically, however, there is no failure-free software. It is therefore important to ensure that tactics are provided to detect

and manage any failures which do occur, in order to avoid any further consequences. **Timestamp** and **sanity checking** are examples of these tactics. It is important to ensure that, where possible, the consequences of failures are managed within the system to avoid any further drawbacks within the system. The **redundancy** and **recovery** tactics help to achieve this.

One of the architectural techniques that explicitly exploits the concept of design tactics is the Architecture Tradeoff Analysis Method (ATAM) (46). ATAM is a structured method for understanding the tradeoffs inherent in the design of architectures for software-intensive systems. This technique was developed as a principled way to evaluate the fitness-for-purposed of software architectures, with respect to many competing quality attributes such as security, modifiability, availability and performance, (85). The technique can help to provide reasoning about architectural decisions which affect quality attribute interactions. Safety can be only one of the competing attributes covered by ATAM and therefore the outcome of the safety analysis should form a core input to ATAM when the method is used for the analysis of safety-critical systems (considering safety but also other important systems and business quality attributes) (58).

2.3.2. Safety Certification

System certification is a very important step and is usually required before the deployment of safety-critical systems, especially if the system can lead to accidents and human injury or death, where "it must be certified as adequately safe according to applicable standards" (86). Ivan defined certification as a process that is conducted by an independent party to assess whether a product, method or service complies with a pre-defined set of requirements (87). The result of the certification process is a written statement, in the form of a certificate.

2.3.2.1. Standards

For most safety-critical domains, different institutions provide standards to guide system development and operation. For example, in the aerospace industry the European Space Agency (ESA) and the National Aeronautics and

Space Administration (NASA) define standards relating to their respective areas. Generally, these institutions are located in different geographical locations (88). The provided standards address safety issues within the system as a whole or sometimes only the safety issues relating to a specific aspect or component of the system (for example, software).

In the nuclear industry, IEC 60880 was introduced by the International Electro-technical Commission subcommittee 54A (IEC SC 54A) to address safety issues related to the introduction of software in current nuclear systems. Prior to that, IEC 60880 addressed only hardware safety in nuclear plants (88). In the UK Ministry of Defence (MoD), different standards have been developed at the software and system levels. The DEF STAN 00-55 standard was developed for software safety and DEF STAN 00-56 was developed for system safety (88).

Kelly (14), Habli (27) and Penny et al. (89) highlight an increased interest in the goal-based approach over prescriptive approaches, based on the reports and recommendations provided from different accidents (e.g. the Cullen report for the Piper Alpha accident).

Generally, there are many standards introduced for the purpose of certifying software safety (90). As discussed below, these standards can be classified either as prescriptive /process-based or goal-based standards.

Prescriptive/process-based

Prescriptive or process-based safety standards define a set of requirements and prescribe the specific means for compliance and the necessary evidence (91). The prescriptive methodology is based on the accumulated experience development and operation of a system in a certain environment. Consequently, when the system is to be located in a new environment or in a novel situation, it will not be adequate to apply the prescriptive methodology (91) (89). Hereña asserts that this drawback will limit the extent to which the standard will be useful.

Technological advances will affect the operations performed within the system. Over time, this leads to new safety requirements, which may not be addressed within the system. Thus, the prescriptive methodology is somewhat limited

when considering time as a factor in the system lifecycle (91) (89). The addition of features and functions to maintain the ongoing safety of the system will contribute to an increase in the cost of the installation and maintenance of the system (91).

A very important issue to consider is legal responsibility. The provider of the system should integrate the necessary safety requirements within the system, and the regulator should ensure that these requirements are accurately integrated. In the case where these requirements did not prevent the occurrence of a certain accident, there is likely to be some dispute as to who is responsible (91).

Goal-based Standards

Goal-based methodologies do not prescribe a process for the development of safety-critical systems, but require developers and operators to provide assurance that the system is safe. In many domains, this assurance is provided in the form of safety arguments that are accompanied by evidence to indicate that the system safety meets the requirements (92). By providing for different arguments and their corresponding evidence, the goal-based approach provides more freedom in developing technical solutions that can accommodate different standards (89).

Kelly, et al. (93) addressed the challenges facing the goal-based approach. Basically, the question that should be answered is "how to comply?". In the goal-based approach, developers produce a set of arguments and pieces of evidence which address the safety issues within the system. One challenge is how acceptable risks should be defined for the system and what is taken to be the accepted level of risk. Another challenge, which is managerial rather than technical, is program management and the costs of developing new systems.

As safety is integrated in the system using arguments and evidence. It is important to establish which arguments and evidence are necessary to satisfy a particular safety requirement (89). Penny et al. argued that the arguments and evidence required are related to the significance of the individual system functions.

2.3.3. Safety Cases

In order for the regulators to be able to certify any system, documentation for the system safety should be presented to demonstrate how the system has met its safety certification requirements. In general, the system developers should define a set of safety goals that the system should meet. For each of these goals (sometimes called claims) an argument and evidence should be presented to support the goal (94). The collection of these arguments and their corresponding evidence is called the safety case of the system (14).

Ideally, the development of the safety case should start as part of the requirements analysis and it should be updated during the different stages of system development. Any later changes in the system should result in a revision of the safety case to assess the impact of those changes on the safety case (89) (14).

A safety case can be defined as:

"*A structured argument, supported by a body of evidence that provides a compelling, comprehensible and valid case that a system is safe for a given application in a given operating environment*." (95)

So the goals, arguments and evidence should address the target environment in which the system is designed to operate.

Each system should have a set of goals or objectives that define its safety requirements within the environment in which it will operate. These goals will be identified based on different factors, mostly derived from the hazard analysis and risk assessment process, as discussed above. As a result, it is logical that these goals can be modified (new goals added or removed) if the system is relocated to a different environment.

Claims about the achievement of safety objectives should be supported by arguments and evidence. Figure 13 shows the role of a safety argument in linking requirements and objectives with specific items of supporting evidence. It explicitly links and justifies how the evidence substantiates the safety requirements and objectives.

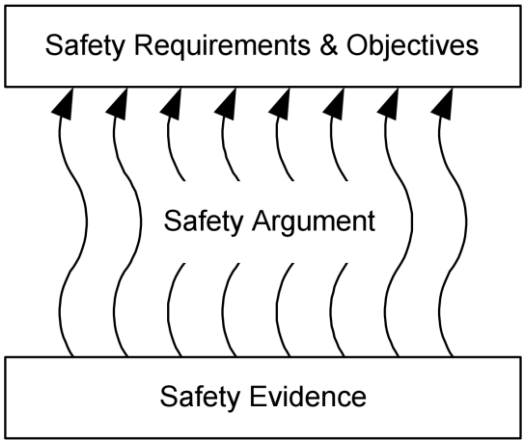


Figure 13: The Role of Safety Argument (14)

One of the aims of the safety case is to evaluate the extent to which the system meets its safety requirements. It should be developed by the system provider and audited by the regulator. Thus, it is important that both parties have a common understanding of the safety case's contents, i.e. the safety goals, arguments and evidence. Given the size and complexity of modern integrated systems, it is expected that safety cases will contain huge amounts of data generated by analysis of the system. In order to ensure adequate presentation of the safety case contents, and to effectively address the system safety requirements, the safety case should be developed using systematic and organised procedures. There is a strong challenge to provide a readable safety case that can be easily interpreted by the regulator (14) (96). Cockram et al. argued that with the huge amount of documentation provided in the safety case, it might be possible to lose concentration on which arguments are provided with which evidence, which makes it hard for the regulator to assess the credibility of the safety case.

2.3.3.1. Safety Case Argument Representation

Safety case arguments can be presented either textually, graphically or in a tabular format. Each of these has its own positives and negatives.

Goal Structuring Notation (GSN)

The Goal Structuring Notation (GSN) is the most widely graphical language in industry for representing safety arguments. GSN was developed at the University of York for defining safety arguments in a graphical format (97).

This notation is expressive and structured to develop safety arguments in a methodical way through capturing the primary argumentation elements.

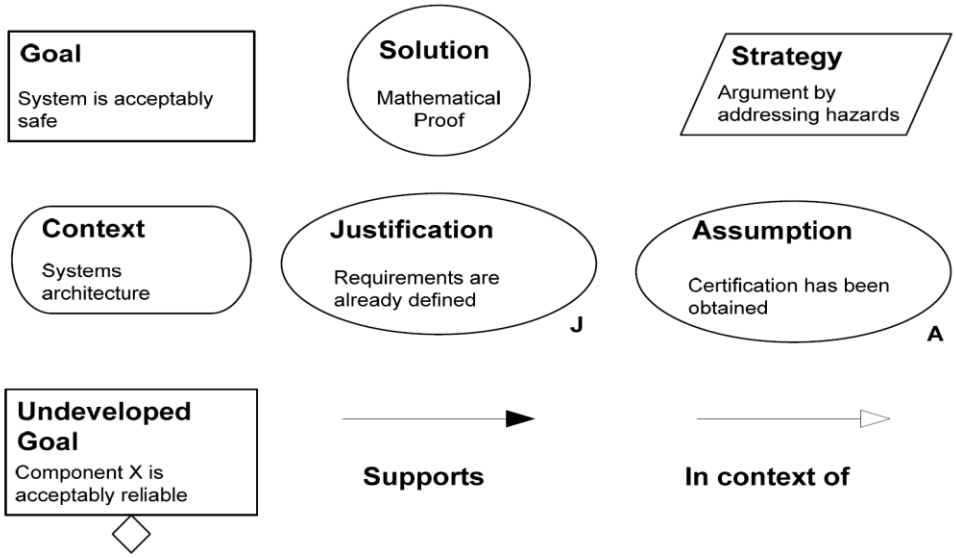


Figure 14: Principal Elements of GSN (14)

Figure 14 shows the basic elements of GSN such as goals, contexts, solutions, justifications and assumptions and their relationships. An argument is created by linking these elements using 'supported by' and 'in context of' relationships to form a 'goal structure (14)'. The goal structure (i.e. claims) can be decomposed into sub-claims and the sub-claims can be further decomposed until these can be satisfied by direct solutions (i.e. items of evidence).

The reasoning behind the decomposition is made clear in the GSN strategy elements. Context elements are used to provide an explicit statement of the context in which the claim is made. Justifications provide reasons why a specific strategy is selected, or a particular claim or solution is used. A claim (GSN 'goal') can be supported by quantitative or qualitative evidence, depending on the nature of the claim. Common types of evidence (solutions) include the results of the hazard identification or risk analysis. Figure 15 shows an example goal structure in GSN, taken from (98).

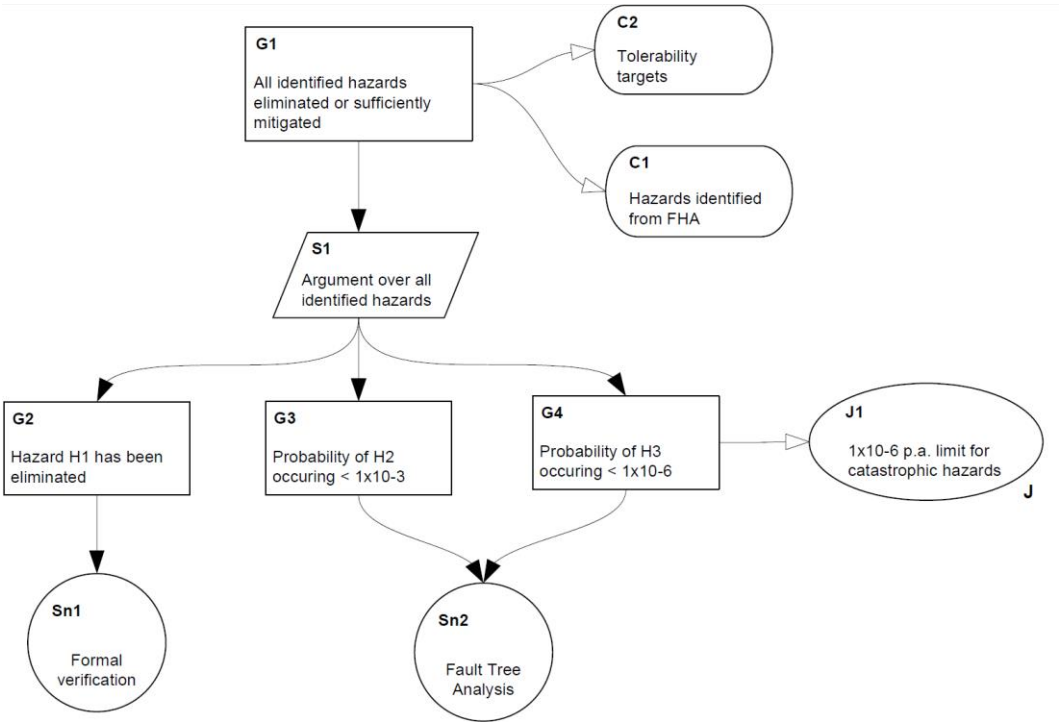


Figure 15: An Example Goal Structure

2.3.3.2. Modular Safety Cases

The modular safety case approach (99) was developed in order to allow the development of a baseline safety case which can be used to reflect additions to or modifications of the system. The approach was developed as a means of managing safety cases for modern, large-scale integrated systems. Furthermore, the modular safety case aims to limit the size of changes in the safety case that are due to the changes in the system structure. In the conventional safety case approach, an independent safety case will be developed for each configuration of the system, which will dramatically increase the time and cost requirements of the certification process. In the modular safety case approach, it is possible to have different variants of the safety case. Each variant is related to one of the system configurations (99).

In the modular safety case approach, a safety case can be partitioned into modules both vertically and horizontally. The partitioning will be based on the top-down progression of the objectives-argument-evidence (100). In vertical partitioning, certain argument claims can be used as objectives for another safety argument, while in horizontal partitioning one argument can provide the assumed context for another.

As different modules are inter-related with each other, it is important to define the dependencies between these modules. Explicit interfaces can be defined between safety case modules. The safety case module interface can be represented as follows (99):

1. Objectives addressed by the module;
2. Evidence presented within the module;
3. Context defined within the module;
4. Arguments requiring support from other modules
   Inter-module dependencies;
5. Reliance on objectives addressed elsewhere;
6. Reliance on evidence presented elsewhere;
7. Reliance on context defined elsewhere.

When developing a modular safety case, the argument contained within a single argument may fail to support all the module claims. Thus, some argument modules can be used provide arguments and evidence to support claims that are recorded in other argument modules**.** However, within any composition, objectives and arguments from different modules should complement each other. Furthermore, the evidence in different modules should not be contradictory, and the context of each module should be consistent with the context of the other modules in the composition.

In compositions, the rely-guarantee approach is used to capture the modules' behaviour within the composition. In the safety case module interface defined above, item 1 will provide the guarantee and items 4-7 will provide the rely conditions. Items 2 and 3 must hold as the composition is being developed.

Graphical extensions for GSN have been provided to handle the modularity of the safety case (99). The first extension to GSN is the representation of the modules, which is a package notation imported from UML. In the module notation goals supported by other modules are represented as Away Goals (101). Further extensions have been developed to allow the link between an argument in a certain module and context and evidence that exist in another

module. To support these links the Away Context and Away Solution are introduced.

In GSN, a safety case module interface is defined as follows (99):

1. **Goals** addressed by the module;
2. **Solutions** presented within the module;
3. **Context** defined within the module;
4. **Goals** requiring support from other modules.

Inter-module dependencies are captured by:

5. Away Goal references;
6. Away Solution references;
7. Away Context references.

The process of composing different modules to form a safe composition should be carefully performed, to ensure that the resulting argument consists of well-defined modules that are correctly connected and interdependent. Three main steps for the composition of safety-case modules are identified by Kelly (99).. These are: Goal matching, Consistency checks and Handling cross-references. At each step, different related issues will be investigated to ensure proper composition of the modules.

While performing the composition of the different modules, documentation for the relations between the different modules should be developed as a safety-case contract (99). This will help in understanding the relationships between the different modules, and to assess whether the composition still holds if the context changes, or if one of the modules is modified or substituted. The contract must record all the away goal and solution references that had been exploited by the composition (99).

The safety case architecture will be similar to the system architecture. The main principles for this are: high cohesion/low coupling, supporting the division of work and contractual boundaries, supporting future expansion, and isolating change (100). Research into the development of Integrated Modular Avionics (IMA)-based systems adapted safety case architectures based on the

modularity of the safety case, which corresponds to the modularity in the system itself (99). Adopting this architecture, the safety case can be divided into modules, and the cross-references between the modules can be represented as well.

In modular safety cases, and based on the principle of isolating change, arguments that are likely to change will be separated in standalone modules. In the case of change, only those modules affected by the change require reassessment; it is assumed that the other modules in the safety case are unaffected. Although it might be possible for the interface of the module to change (which requires changes to the contract to be made), the paths that require revision in the safety case can be identified easily. The changes should be limited to the specific module in which the change occurred and the interfaces connected to other modules within the safety case.

The final architecture of the system and its corresponding safety case should be evaluated to ensure that they meet the architectural quality attributes, such as scalability, performance and modifiability. Different evaluation techniques have been proposed, either for the system as a whole or for the software element of the system. An example is the Software Architectural Analysis Method (SAAM) (102).

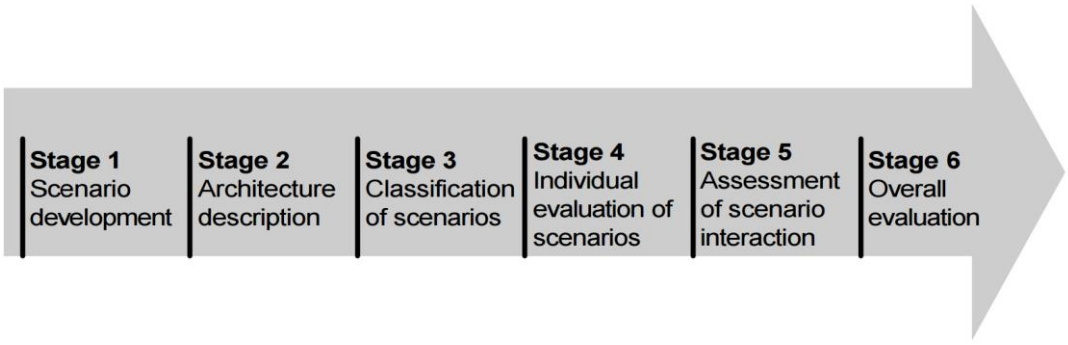


Figure 16: Activities in SAAM Analysis (102)

SAAM proposed a systematic evaluation process for the system architecture that is modular, as shown in Figure 16 (102). Each step in the method performs the necessary evaluation for a certain module or set of modules within the architecture, trying to identify the required changes to the architecture and the effort required to enact these changes. Generally, changes will be at one of

three levels: local to a certain module within the architecture, for a set of modules within the architecture, or for the architecture as a whole.

GSN: In our work we use GSN to develop SOA safety cases using the modular and pattern extensions of the notation. We applied GSN for defining argument modules and patterns which capture the different argument structures for assuring services and their design and constrain variation in SOA safety cases.

Modular safety cases have been adapted for different applications. For example, they have been used to provide assurance for Integrated Modular Avionics (IMA) (103), for operating systems (104), for Objected-Oriented systems (26), for COTS (105) (106), for Kernels (107) and for product lines (27).

2.3.3.3. Modular and Incremental Certification

The introduction of the modular safety cases technique inspired proposals to concerning the adaptation of the certification process. By limiting the effects of change within the safety case, and safety assurance work, modularity theoretically allows for the limited certification of certain system elements: the effort required to certify changes from a previously-certified baseline in a given composition should be smaller than re-certifying the entire system. By adopting the same modular and incremental approaches, the certification process simulates the modularity of the safety case (108). Thus, in theory, the re-certification of a system due to any change or addition to the system should only require the assessment and evaluation of the changes, which will limit the re-certification effort to the size of the changes, rather than to complete recertification of a complex system.

Rushby (109) proposed an approach based on the concept of assume-guarantee reasoning. By investigating the presence of failures, the approach will extend from verification into certification. However, the approach requires the use of partitioning, whereby a single module can be reused within the system, or can be deployed in another system. This approach is based on the modular certification process proposed for IMA (103), where certification of the system can be performed based on a modular safety case, which facilitates

the certification of the changes in the system, by focussing on certification of those modules involved in the change and their interfaces.

In traditional certification, the cost of the re-certification is related to the system's size and complexity, as recertification typically requires the development of a new safety case, in total, in response to the system changes. With the introduction of modular safety case and certification approaches, it became possible to relate the cost of the re-certification to the size and complexity of the changes, which will dramatically decrease the total cost of re-certification, especially in the case of small changes to large systems (110).

The project conducted by IAWG investigated the construction of modular safety cases and proposed a methodology for incremental certification in the UK defence avionics domain. In the case of any changes to the system or its proposed operating environment or use, evaluation of these changes will define where changes are also required in modular safety case relating to the system. In most certification regimes, any changes to the system safety case require a re-certification process to be performed. In the case of modular safety cases, the re-certification will be only performed on the parts of the safety case that have actually changed, or which address aspects of the system design which have changed. The process is called an incremental certification process. Fenn et al. (108) assert that the incremental certification process is still immature.

The IAWG modular certification process defines a series of systematic steps to assess where change is required in the safety case and what concerns must be addressed. These steps start by identifying the change scenarios, then performing optimisation for the safety case architecture, followed by identifying the Dependency-Guarantee Relationships (DGR), and finally building the safety-case arguments. These steps are discussed in detail in Fenn et al. (108).

Although the modular and incremental certification processes are expected to have benefits over traditional certification processes, these approaches cannot be considered suitable in all cases. As part of the IAWG project, five key criteria were identified to ensure that applying modular and incremental certification approaches are likely to be beneficial in a given case. The five keys are: distilled

set of change scenarios, re-use, and modularity, use of COTS and vendor co-operation, and size and complexity of the system (108).

2.3.4. SOA and Safety

For an SOA system that is considered to be safety-critical, or has any safety-related impact on humans in particular or on the environment in general, a certification process is often conducted to ensure that the system is safe to operate in its designated context.

In Section 2.2, we discussed SOA designs. The discussion included the SOA lifecycle, tactics, design patterns and system modelling. In Section 2.3, we discussed the different issues related to the safety of the system. This includes defining the safety requirements, developing the system safety case and the certification process.

The aim of this section is to discuss published work related to SOA safety. In the following, a review will be carried out on existing work on SOA applications for safety-critical systems and safety assessment.

Existing work on SOA has lacked sufficient consideration of potential safety applications. Only one paper could be found in the literature (16) on the use of modular safety certification for SOA. However, there is published work which touches on some safety aspects of SOA.

Donini et al work focused on testing safety-critical SOA-based systems through the adoption of a classical SOA to build an environment for safety-critical control systems (111). They provided indications on the integration of SOA system architecture components with existing centralised testing environments by addressing concrete test objectives. Specifically, they provided a test framework where automated functional testing is presented in an external simulated environment based on SOAs. They also conducted functional system testing via simulated environments which can be an approach to minimising test sets. The test sets defined were representative of an actual operational usage profile for a railway interlocking control system.

Rychlý has created an approach to representing the behaviour of services within SOA and their implementation through algebra π-calculus (112). This is

represented by using the case study of Donini et al (111) based on a railway interlocking control system which is driven by an SOA design (113). Detailed description of the SOA and all its services is introduced in (112).

The researchers in (114) proposed a framework for using SOA in non-critical parts of safety-critical embedded systems. This allowed the use of Smart Sensor Interfaces (SSI) and Smart Sensor Protocol (SSP). SSI can facilitate the integration of the payload on a UAV represented by sensors. SSP can be used to exchange information to decide the requirements for mission accomplishment (114). Based on an SOA design, the paper demonstrates that the real-time performance is compulsory and has to be guaranteed under different circumstances.

In (115), the authors proposed a Knowledge-Based Framework for Dynamically Changing Applications. The framework uses the dynamic definition of web services, based on a specific system configuration, the availability of the web service component and the database with predefined context information. The framework allows the monitoring of the information to generate services with different levels of security, reliability as well as performance, in a dynamic way. Nevertheless, the implications of SOA systems for safety certification are not well explored.

The above pieces of work are different from our work. Although they consider safety-critical systems, they do not cover any aspects of safety analysis and safety cases.

The researchers in (116) have developed a methodology for risk management within SOA to address risks, threats and vulnerabilities, through using the Operationally Critical Threat, Asset, and Vulnerability Evaluation (OCTAVE), National Institute of Standards and Technology (NIST) and Factor Analysis of Information Risk (FAIR). These techniques can be used to provide assurance to changes within safety requirements (116). Similar to our work, they produce safety requirements at different stages of development. They present a development model based on five SOA layers where each layer is composed of phases. These phases have activities, which can be subdivided into tasks, which are the subject of the safety analysis. Their work is similar to our work as far as

safety analysis is concerned. However, it does not cover any aspects of safety cases or any interfaces between design and modelling, and safety analysis.

Brown et al have addressed the issue of assuring SOA designs using modular safety cases. Similar to our work, they used GSN to create these safety cases (16). They have raised some challenges to safety when using an SOA style. These are related to:

- System/service requirements
- System safety certification
- Safety certification throughout the system development lifecycle.

They introduce the problems and propose a high-level methodology for creating a safety case within the military aviation domain. Brown's argument is meant to support the flexibility of compositional features in SOA, while simplifying the safety certification and accreditation of SOA-style systems (16). However, their work was limited in scope, focusing on an example application rather than creating an integrated framework for modelling, analysing and assuring the safety of SOA in safety-critical applications.

2.4. Chapter Summary

In this chapter, we reviewed the definitions, features and lifecycles of SOAs. We discussed the SOA system architecture and gave an overview of the core elements of SOA covering the SOA design patterns and notations used for SOA modelling. We introduced the concept of system safety, and introduced safety-related issues and certification processes and discussed the most common elements of concern in system safety engineering, such as the overall system safety process, safety analysis, risk assessment, risk management, safety requirements and safety tactics. In addition, we explained techniques which are commonly applied to address these elements. Then we moved on to discuss safety certification and safety standards. We considered the concept of safety cases in some detail addressing approaches to safety-case development, challenges for the safety-case approach, and argument representation techniques. Finally, we reviewed the concept of modular safety cases and its application for modular and incremental certification.

Chapter 3

3. SOA Safety Assurance Framework

3.1 Introduction

The literature review in Chapter 2 covered key issues related to the development and assessment of SOA. In particular, it considered the safety and dependability properties of SOA and related architectures, such as cloud computing and SoS, used in critical industries. Based on the review of current literature concerning SOAs and safety-critical systems, the following observations can be drawn:

- There is a shortage of research on the use of SOAs for the development of safety-critical systems.
- There is a lack of systematic safety assessment processes and methods that address the special features of SOAs such as composibility.
- There is a lack of safety assurance strategies in the form of safety argumentation and evidence to justify why the use of SOAs in safety-critical applications is acceptably safe.

In this chapter, we introduce and define a conceptual framework for SOA safety assurance. This framework is discussed in the context of architectural modelling, safety analysis, safety certification and tool support, and methods to support the framework are proposed. These methods are explained in more detail in Chapter 4 (SOA Safety Analysis) and Chapter 5 (SOA Safety Cases).

3.2 Conceptual Framework

The main aim of the SOA safety assurance framework presented in this chapter is to provide a systematic means to understand and analyse how SOA can contribute to safety and hazards and how safety justifications can be provided by means of structured and explicit safety cases. The framework is summarised in Figure 1.

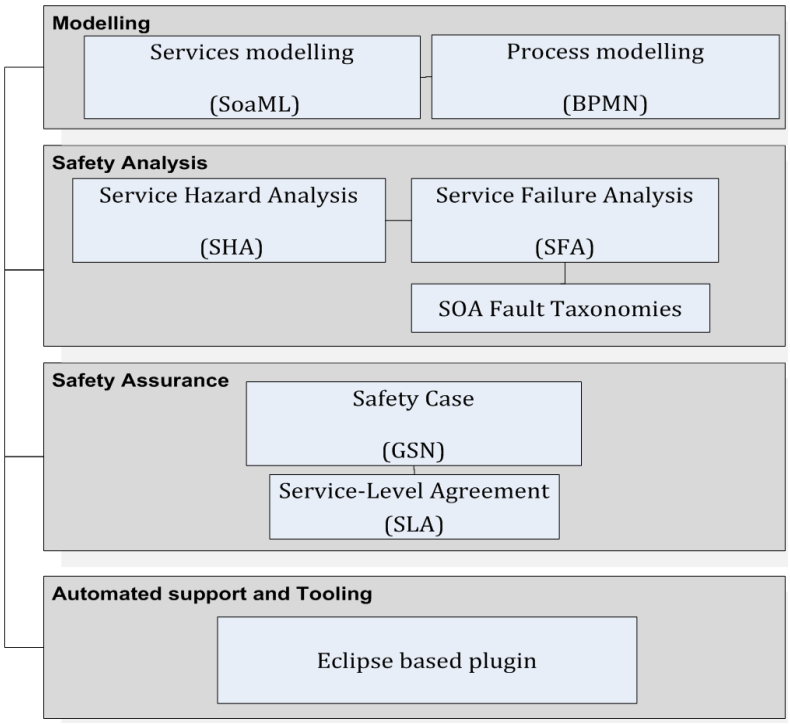


Figure 17: SOA Safety Assurance Framework

Figure 17 shows the four aspects that have been considered and developed in order to address key concerns about SOA safety: modelling, safety analysis, safety cases and automated support and tooling:

- **Modelling** is used to represent the structure and behaviour of the service-based system, developed against a variety of design styles. This covers the design of the services, contracts and interactions and the specific tasks and flow of information used for realising the SOA design.
- **Safety analysis** results are used for understanding the failure behaviour of the SOA and specifying safety requirements and design tactics to control the risk of any hazardous failures. The framework adapts and develops two safety analysis techniques for understanding service

hazards and examining how the system failure behaviour can contribute to the occurrence of these hazards. In early safety analysis, engineers designing services used in safety-critical applications need to define safety requirements that address the risk of any hazardous behaviour posed by these services and ensure that the design of the SOA explicitly meets these requirements.

- Justification of the safety of the service-based system is communicated through the development of an explicit **safety case**. A safety case is a structure that comprises an *argument* and *evidence* that present the reasoning for why that system is safe for a given application in a given environment (99). In our framework, the safety case is based on the results generated from the SOA models and safety analyses.
- The **tool-support** environment provides automated capabilities for building the SOA models, applying the safety analysis and developing the safety case for the system. The tool environment, developed during this PhD research, also provides means for ensuring traceability between the modelling, safety analysis and safety case artefacts.

In short, the SOA safety assurance framework proposed in this thesis provides a methodical basis for integrated design and analysis of service-based systems, with explicit consideration of hazardous behaviour and safety justification.

3.3 SOA Modelling

This section introduces the modelling languages that are used in our framework for specifying the services and the dynamic processes for SOA-based systems. These modelling languages are employed in order to provide a holistic view of the design elements of the SOA within the system, considering structure and behaviour which are important for the safety analysis, and to generate models which can be understood by stakeholders with different levels of technical background.

In terms of the structure of the SOA, the types of concern covered by our modelling include:

- Services

- Interfaces
- Contracts
- Operations
- Actors

The modelling language we use for specifying the above structures is SoaML (117), which was reviewed in Section 1.7. SoaML models are based on a modular description of the SOA in which related *Contracts*, *Interfaces* and *Operations* are encapsulated in, and provided by, self-contained *Services*. SoaML allows for the specification and analysis of the requirements for each *Service*, including the safety requirements, which is an important feature of the language for dealing with safety-critical services. Further, SoaML can provide an explicit description of *Services* in terms of actors, or *Participants*, which realise and implement these *Services*.

The proposed framework models SOA behaviour, including behaviour that can lead to hazardous events, using BPMN to describe the processes and the information flows in the SOA design. BPMN captures the dynamic *Tasks* and *Flows* implemented in the SOA design. BPMN allows the modeller to communicate and represent internal procedures, based on processes, in a graphical and structured notation. In our framework, BPMN is used to understand and analyse the detailed failure behaviour in a service-based system and how this can contribute to service hazards. However, the conceptual framework can be seen as a notation-independent solution. Any notation which has sufficient power to model the structural definition of services and their behavioural implementation could be used as part of the framework. There are many alternative languages which could reasonably be used: each modelling language will have subtly different approaches, but the concepts required by our framework are sufficiently generic that the choice of language does not affect the application of the framework. In this thesis, SoaML and BPMN are used as the two main modelling languages in order to illustrate more concretely the concepts and the methods that are developed as part of the framework, and to show how traceability can be achieved between the

structure and behaviour of the SOA design (as an input for the safety analysis and safety case methods).

3.3.1 Modelling SOA Services in SoaML

As stated above, SoaML is intended to model the structure of the services that comprise an SOA-based system in terms of *Participants* (*Consumers* and *Providers*), *Services Architecture*, *Service Contracts*, *Service Interfaces* and internal *Operations*. It is focused on the integration of Participants into the *Services Architecture* model using Service Contracts, which capture the *Service-Level Agreements* (*SLAs*). In our framework, this level of detail is important for understanding how services can be hazardous and how Contracts, Interfaces and Participants can contribute to service hazards.

In addition, SoaML can help identify and specify policies for providing and using the services and classification schemes. We use this to support the definition of severity classification schemes when applying SOA safety analysis and their effects on the safe operation of the system.

The SoaML specification describes the services provided by the architecture, including different areas that can be useful for safety analysis. Participants are used to model the service *Providers* and *Consumers* within the system. This can help in understanding how service Providers and Consumers can potentially deviate from the intended behaviour and contribute to unsafe failures. The interactions between Participants are through service *Contracts*. Service Contracts are used to explain the interaction among service entities. Like Participant failures, communication failures can also contribute to unsafe failure behaviour, via deviations from the intended Contracts.

Service *Interfaces* are used to define the *Operations* that are offered by Providers and are available for Consumers in an SOA. The technical Interfaces include signatures of Operations that include the names of the Operations, the parameter types and the return types of the Operations. Since Interfaces are used to specify the input and output values for the processes, failures such as the loss of data or the provision of incorrect descriptions of data via these Operations could lead to unsafe behaviour of the system. These failures could

also delay the Operations within the system and might contribute to service hazards.

Services Architectures are the diagrams used to describe the main components and communications within the system and how Participants work together using Services. Contracts are used to display the relationships between Participants and Services. These relationships can help to identify the effects of failures within the Services and Participants, clarifying the level of severity and deriving the safety requirements that are specific to the Services.

In summary, SoaML provides a structured basis for modelling the high-level services and their interactions, including interfaces. This is necessary to perform hazard analysis on service-based systems.

3.3.2 Modelling SOA Processes in BPMN

The framework uses BPMN as the modelling notation for specifying dynamic processes in the SOA design. The BPMN model specifies the behaviour of the processes underlying the services and identifies *Tasks*, such as *Script Tasks* and *User Tasks,* as well as *Sub-Processes*, *Gateways, Sequence Flows, Pools and Lanes*. It also describes how messages are exchanged and coordinated between Participants. The diagram describes the execution of processes within an SOA system and provides a key input for understanding both the intended and failure behaviour of the system. This is a key prerequisite for performing detailed safety analysis of the low-level behaviour of services and their interactions.

A key concern in SOA safety is understanding how interaction failures can lead to hazards. BPMN models focus on the orchestration of tasks that implement high-level services in business processes. This includes specification of the choreography and collaboration between interacting processes. Defining and understanding these interacting processes is key for analysing the failure behaviour of the services.

Another area of concern for understanding the safety of system behaviour is the flow of information and control within the SOA. Service-based systems tend to be information-intensive: key decisions are based on the information

processed by the system and provided by different services and participants. Analysing the failure modes associated with information flows is necessary for analysis of the safety of the SOA system, and BPMN provides a structured means for capturing the flow of information and understanding the consequences of any potential deviations.

The BPMN process defined in our framework can help engineers to focus their analysis of the low-level service behaviour under different situations and timing conditions. Each process captures the tasks that should be done by the identified service Participants and the coordination between them. The timing of messages between processes has to be scheduled, in order to clarify the overall duration for each process, or the worst-case timing behaviour. This enables the analyst to see how timing failures can contribute to service failures. Some of these failures might be hazardous.

In summary, BPMN provides a detailed representation of the execution mechanisms underlying the high-level services and is used as a basis for the detailed failure analysis of the service-based system behaviour.

3.3.3 Traceability between SoaML and BPMN Models

In our framework, the SoaML models, used for defining the services, and the BPMN model, used for specifying low-level behaviour, are linked explicitly at the model level.

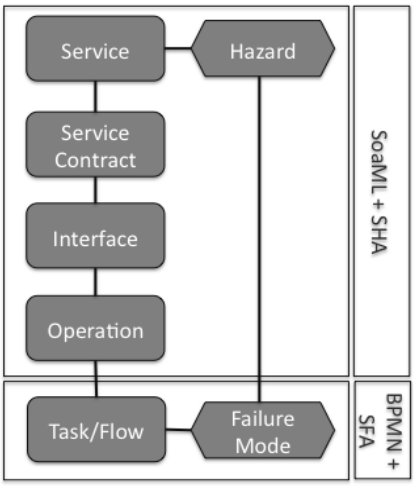


Figure 18: Link between SoaML and BPMN Models

As illustrated in Figure 2, Services in SoaML interact through defined Contracts. These Contracts have explicit Interfaces that offer a number of Operations. We link each of these Operations to a Task in BPMN, thus enabling traceability between the different models at different abstraction levels. Further, we represent every Participant in SoaML as a Swimlane in the BPMN model. A link between two BPMN Tasks should capture the information flow in terms of inputs, outputs, data, sequence and timing. The explicit traceability between the various SoaML and BPMN models is necessary for assuring consistency in the SOA safety analysis and safety cases. The interaction between the SoaML and BPMN models is explored in detail in the next section.

3.4 SOA Safety Analysis

This section introduces the two safety analysis techniques we propose for use in hazard and failure analysis for SOA-based systems. In the design phase, safety analysis is used to derive the necessary safety requirements that define the basis for ensuring that the SOA system developed is acceptably safe. Safety analysis is carried out as part of a safety management process, which forces explicit consideration of the system safety level and the system deployment. The analysis allows for identification of the system-level hazards associated with the services and their interactions and of the potential failure modes that can contribute to these hazards.

The consideration of hazards and failures is a fundamental issue for service-based applications used in the safety-critical domain. It is important to implement the analysis of the hazards, identify them, evaluate their effects, identify the severity of the hazards and then provide the necessary measures to manage the service hazards. The service hazard analysis technique proposed in this thesis helps to identify the top-level conditions that may lead to unsafe system events.

The safety process should be designed to address factors within the system and its environment which potentially affect the design and operation of the service, e.g. the size of the company, the structure of the system, the industry the system is operating in and the applications the system is used for (Leveson, 1995). A safety process will have different activities, depending on the SOA

system that is being investigated, with specific reporting requirements for each activity. Each activity will result in a detailed documentation of results. Applying safety analysis to an SOA system should determine the safety requirements for the services and processes and provide evidence for the processes of assuring that the system is acceptably safe. In addition, safety analysis will affect the ability of the system to be deployed.

The framework uses and adapts two existing safety analysis techniques, namely Functional Hazard Assessment (FHA) (118) and Software Hazard Analysis and Resolution in Design (SHARD) (72) and tailors them to the analysis of individual services and SOA processes. The aim of these techniques is to provide a systematic basis for identifying and analysing hazardous behaviour associated with the SOA system (e.g. hazards due to lack of information, or due to incorrect, information). The system safety team can thus decompose high-level safety criteria into readable and detailed subsets, which can be used by the design team to develop the SOA system (73). Safety analysis should also be performed to assess the risk associated with each of these service hazards and to identify measures for managing the risk.

The rest of this section considers the safety analysis of both services and their underlying behavioural processes, considering service hazards and the failure modes that can contribute to these hazards.

3.4.1. Safety Analysis of Services

The framework proposes to perform safety analysis for services modelled in SoaML, through adapting FHA as a means for identifying and classifying service hazards. A service hazard is a condition associated with the service, or its interactions, that can lead to an accident. This thesis defines an adapted FHA technique for identifying and analysing service hazards, which we have called Service Hazard Analysis (SHA). SHA addresses individual services, considering their associated hazards, the effects of these hazards at different levels and the severity classification of these effects. Based on these factors, SHA provides suitable actions or recommendations for creating safety requirements for the service.

In terms of the "V" safety lifecycle model discussed in Chapter 2, we adopt this technique to consider early lifecycle hazard and risk assessment. At this stage of the conceptual design, the analyst has to consider broad hypothetical failure types, since the system has not yet been implemented and detailed hazard analysis is therefore not possible. These failure types are generic and consider issues such as the complete absence of the service, unintended service provision or incorrect service operation. Each of these failure types will have to be interpreted and classified in the specific context in which the system is used.

In summary, in our framework, SHA provides a list of service hazards, their safety classification and a set of requirements needed for managing those hazards through subsequent design, development and deployment activities. SHA is described in detail in Chapter 4.

3.4.2. Safety Analysis of SOA Processes

In order to identify and assess the design failure modes that can contribute to service hazards, we have adapted the SHARD technique to provide a means for understanding detailed service failures. We call this technique Service Failure Analysis (SFA).

SHARD, which provides the basis for SFA, is one of several software and hardware safety analysis techniques which address information and control flow and intended behaviour and how failures within information flows in particular can be lead to hazards (72). SHARD is a variant of the hazard and operability study (HAZOP) technique (119). In our framework, SHARD has been adapted to analyse the flow of information between the Tasks represented within the SOA processes (captured in the BPMN models). The analysis identifies a series of failure modes, and is driven by the application of a set of guidewords to each information flow. The guidewords are *omission*, *commission*, *early*, *late* and *value*. These guidewords are used to define the types of failure modes that may occur during the execution of the SOA process. Each of these failure modes is then associated with specific service faults linked to existing SOA fault taxonomies (i.e. potential causes). It then identifies the effects of the flow failures. The output of this analysis is a set of derived safety requirements.

More specifically, SFA starts by defining the deviations that could potentially be exhibited in behavioural SOA processes. These deviations are identified by applying and interpreting the abovementioned guidewords against information flows defined between the Tasks modelled in BPMN. To make the analysis specific to SOA failures, the causes of each potential deviation is linked to a specific type of fault as defined in the SOA fault taxonomy used in this thesis (Chapter 4). Next, the impact of each flow deviation is assessed in terms of the contributions that it could make to the service hazards (i.e. as identified in SHA). Finally, SFA produces detailed safety requirements and design measures for potentially meeting these requirements i.e. in the form of specific SOA design tactics. These tactics are based on existing safety tactics in the software architecture literature or SOA-specific dependability tactics (e.g. which are centred on the use of service redundancy, diversity, graceful degradation, monitoring and containment) (120).

3.4.3. Use of SOA Fault Taxonomies in SFA

A typical SOA has five major steps in its execution, namely: service publishing, service discovery, service composition, service binding and service execution. Each of these steps may be exposed to faults when running an SOA-based system. If faults are not detected, they may lead to failures at the service and system level (some of which are hazardous). The SOA literature covers ranges of faults identified and categorised by the community (i.e. SOA-specific faults rather than general system faults). One of the most thorough fault taxonomies for SOA was defined by Bruning, Weissleder and Malek (121). This taxonomy is well structured and categorises faults based on the SOA lifecycle phase. Fault types in the fault taxonomy are used in our framework as prompts or hints for safety analysts when performing SFA rather than as an exhaustive list of all possible SOA faults.

3.5 SOA Safety Cases

Safety certification often requires a safety case to be constructed in parallel to the development of a safety-critical system (14). In this section, we discuss the approach we developed to constructing SOA safety cases using the modular and pattern extensions of GSN (99). We use the Goal Structuring Notation

(GSN) (97) to define modules and patterns which capture the different argument structures for assuring services and their design and constrain variation in SOA safety cases. The modular safety case and catalogue of argument patterns we have developed for SOA are introduced in the next two sections and described in detail in Chapter 5.

3.5.1. Modular Safety Case Development for SOA

The SOA safety case architect plays an important role to put together information regarding the safety conditions and safety requirements, based on the SOA safety analysis described in Section 3.4 above, in order to capture the safety reasoning and provide assurance about the safe behaviour of the SOA-based system. In order to improve traceability, it is desirable for a safety case to have a clear correspondence with the design of the system. Essentially, an SOA is a modular architecture, where each service represents a separate high-level module with defined interactions. In this thesis, we adopt the notion of modular safety cases and examine how modularity in the assurance case can correspond to the modularity in the design of the SOA.

More specifically, the Goal Structuring Notation (GSN) is used to represent a series of safety argument modules (93). A GSN argument module is a self-contained structure which represents the argument relating to a specific aspect of the safety of the system and the evidence which supports the argument. A modular safety case comprises several linked argument modules, each concerned with a discrete element of the system, which combine to provide assurance for the system as a whole. Modularity in the definition of services in SOAs lends itself to the concept of modular safety cases. Services and participants are self-contained system elements that interact through predefined interfaces (e.g. provided through SLAs). Service owners or producers often rely on other services when guaranteeing and providing their own services (122). Similarly, claims in certain argument modules can only be said to be substantiated (the guarantee clause) if claims or evidence are available in other argument modules to offer sufficient support (the rely clause).

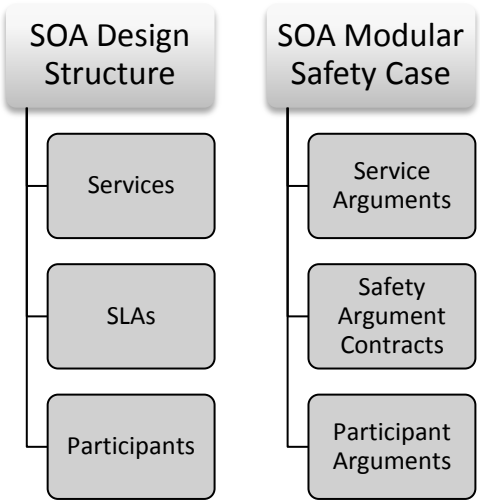


Figure 19: SOA and Modular Safety Case Correspondence.

Figure 19 gives an overview of the key modules in the design and safety case of an SOA, as considered in this thesis. We enforce a traceable relationship between the high-level SOA design, as captured in SoaML (*ServicesArchitectures*), and the overall SOA safety case, as defined in modular GSN. For each service in the SOA, we create a service argument that addresses the hazards posed by the service (based on the results of SHA), i.e. the argument provides a justification for how the service hazards have been managed. Similarly, for each participant, we create a safety argument that justifies the contributions that the participant makes to service hazards (based on the results of SFA). Furthermore, participants interact through defined SLAs in order to provide and consume services. These interactions between participants can contribute to safety and therefore their assurance is addressed in explicit safety argument contracts.

3.5.2. Pattern Catalogue for SOA Modular Safety Cases

As outlined in Section 3.4, the safety analysis of SOA follows a structured, though implicit, line of reasoning: an SOA-based system is acceptably safe because all credible service hazards have been identified and managed. This is supported by the evidence from the analysis of services and their interactions provided by SHA. Furthermore, this line of reasoning is supported at a lower level through the analysis of the technical failures that contribute to the service

hazards. This is supported by the results of the analysis of tasks and flows provided by SFA.

In this thesis, we propose a set of argument patterns to capture this line of reasoning (123). The main objectives are to ensure consistency across modular arguments and to allow the safety analyst to reuse the argument patterns as a basis for the safety assurance of different SOA-based systems. The argument patterns catalogue we propose for SOA safety cases (99) is presented in Chapter 5.

The SOA safety argument patterns catalogue provides framework argument structures to address aspects of the safety of using an SOA system within a specific safety application. Although the argument patterns in the catalogue are separately defined, they are directly related to each other to inform the development of an overall modular argument structure. Therefore, the SOA argument patterns catalogue includes rules for pattern composition and traceability to the architectural design (i.e. SoaML and BPMN models) and analysis (i.e. SHA and SFA).

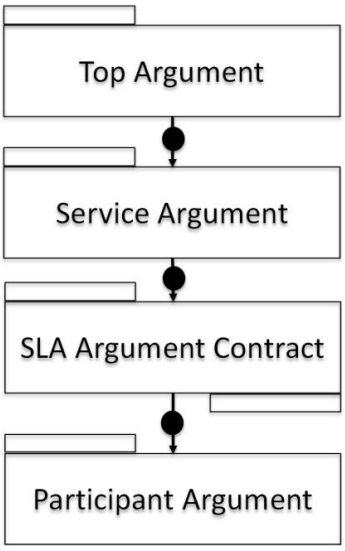


Figure 20: SOA Argument Patterns Catalogue

The SOA safety argument pattern catalogue developed in this thesis consists of three individual arguments and one argument contract (Figure 4). These patterns are established according to an overall hazard-directed argument (Top Argument), which is supported by a number of Service Arguments, SLA Argument Contracts and Participant Arguments. These patterns are composed

together to form a complete justification regarding the safety of the overall SOA system design. The pattern we have defined enable us to present the argument regarding the safety of the SOA system from the perspectives of both the Services and the Participants. The SOA safety argument pattern catalogue is explained in detail in Chapter 5.

3.6 Automated Support and Tooling

This section presents an overview of the automated support that was developed for the safety analysis and assurance of SOA and the tolling environment used to implement it. This covers the three main elements of our SOA safety assurance framework: modelling, safety analysis and safety cases. Figure 5 shows an overview of the integrated tools that have been used to support our SOA safety assurance framework. The main capabilities of the tooling environment are as follows:

1. Create SoaML and BPMN models
2. Create and integrate SLAs into the SoaML models
3. Integrate BPMN and SoaML models
4. Execute BPMN models
5. Embed SHA results into the SoaML models
6. Embed SFA results into the BPMN models
7. Create GSN-based arguments and patterns

In order to represent SOA, we implemented most of the models in Eclipse (124). Eclipse is an open source Integrated Development Environment (IDE) platform that provides support for creating, deploying and managing software, and which can be extended in terms of frameworks, runtimes or tools. For service modelling, the SoaML plugin is the tool used to model SoaML inside Eclipse. It was extended from Papyrus (125), which is an Eclipse plugin for creating UML models such as SysML, UML and UML profile. For BPMN modelling, Activiti Business Process Management (BPM) was selected. Activiti BPM is an open source and lightweight workflow platform for BPMN that supports BPMN 2.0 specification for modelling and execution. (126).

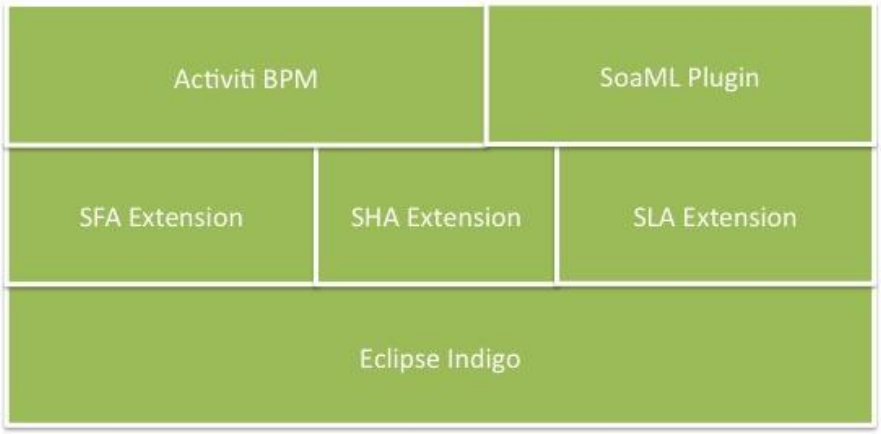


Figure 21: Overview of SOA Safety Assurance Tools

To create a business process, Activiti BPM provides solutions based on an Eclipse plugin and a web-based modeller called Activiti Modeller. The Activiti Engine can be used to make the business process executable. Moreover, Activiti has an Activiti Explorer which is a web application for testing or simulating instances of business processes. These executable features are useful for the analysis of the dynamic processes underlying the SOA architecture.

In order to support the SOA Safety Assurance Framework presented in this thesis, we implemented an integration basis between SoaML and BPMN models in order to improve the traceability between the design of the services (mainly structure) and their underlying processes (mainly behavioural). We also extended the Activiti modelling environment with capabilities to support the documentation of the SLAs for services and the results of SHA and SFA by providing automated means for displaying the safety analysis for both services and flows between tasks.

3.7 Summary

In this chapter, we introduce a conceptual framework for SOA safety assurance and the tooling we have developed to support it. The framework defines how SOA system modelling, safety analysis and safety certification captured in such a way that they can be integrated with each other. The chapter introduces the modelling languages and approaches that the framework uses for specification of the services and dynamic processes in SOA-based systems. It introduces methods for identifying service hazards and potential failure behaviour in an

SOA system design. It also proposes the use of GSN modules and patterns to define the different argument structures for assuring services and their design and constrain variation in SOA safety cases. The last section presents an overview of the automated support that was developed and the tooling environment that was used for the safety analysis and assurance of SOA. Subsequent chapters of this thesis examine the elements of the framework in more detail. In the next chapter, we consider the two SOA safety analysis techniques that this thesis has adapted: SHA and SFA.

Chapter 4

4. SOA Safety Analysis

4.1 Introduction

In this chapter, we propose a process for the safety analysis of SOA that is integrated within the engineering of service-based systems. The safety analysis techniques presented here capture and examine hazards and failure behaviours relating to services and tasks, respecting the ways in which these services and tasks are defined in the SOA context. The techniques are illustrated using a real-world case study from the healthcare domain.

The SOA safety analysis is an essential stage in the definition of the SOA safety requirements. Similarly, producing these requirements successfully can provide useful input to an integrated and verifiable approach to assuring the safety of the SOA system.

This chapter presents two safety analysis techniques for examining an SOA design and generating the necessary safety requirements: Service Hazard Analysis (SHA) and Service Failure Analysis (SFA). The safety requirements generated from this analysis provide the basis for integrating safety considerations into the design of the system. For example, issues raised during the analysis can be used to inform the choice of of architectural tactics used in design. The analysis also emphasises the interactions between services and tasks in order to identify failures in the behaviour of the interactions which could contribute to service hazards within the service-based system, and thus potentially to accidents. The results of these safety analyses should help prevent the occurrence of failures and hazards by defining and enforcing a set

of Service Safety Requirements (SSRs) at the service level and Derived Service Safety Requirements (DSSRs) at the more detailed design level (e.g. tasks and flows implementing services).

As described in Chapter 3, this thesis proposes a framework for modelling the design and analysing the safety of SOA systems and for providing assurance that the system is acceptably safe. In this chapter, we consider the identification of potential safety deviations that are of interest in an SOA-based system and suggest suitable mitigation design measures for addressing these deviations.

This chapter is organised as follows. Section 4.2 introduces a case study, which is used to illustrate and evaluate the SOA analysis methods we propose. Section 4.3 provides a detailed explanation of SHA, with reference to its application in the case study. Section 4.4 discusses SFA and how it has been applied in the case study. Section 4.5 describes the implementation of these SOA safety analysis techniques in the tool-suite developed in this research.

4.2 SOA Healthcare Case Study

An SOA provides a "*paradigm for defining how people, organizations, and systems provide and use services to achieve results*" (2). For healthcare services, the SOA paradigm can assist in the assessment of patient safety, in which accidents predominantly occur as a result of a combination of human, technological and organisational failures (127). SOA-based systems can lead to people being harmed or property being damaged. For example, patients might be harmed due to loss of their health records.

The first case study undertaken for this thesis was the analysis of part of an ambulance system. The case study includes a general overview of the ambulance and Electronic Health Record (EHR) services in the healthcare domain. The system focuses on one medical pathway that considers the processes and services that are used to transport pregnant women. These processes and services were adapted from the UK National Health Service (NHS) Map of Medicine (128). The ambulance system is based on the description provided in (129) (130). In this study, we have performed

additional modelling beyond that presented in in this Chapter (please see Appendix A). Also, we have not covered other medical processes which are out of the scope, e.g. childbirth services.

The objective of this case study is to provide rapid feedback on the techniques we have developed, and to also illustrate the application of the safety analysis methods and safety case approach. The case study is concerned with three categories of service: ambulance, electronic health records and childbirth services. These have been used to provide 10 detailed services within the system, which are as follows:

- Request Ambulance
- Dispatch Ambulance
- Update Status
- Request Record
- Provide Patient Record
- Update Health Record
- Retrieve Health Record
- Pick up Patient
- Examine Patient
- Drop off Patient.

The following subsections provide an overview of the three main categories of service used in this case study.

4.2.1 Ambulance Services

Ambulance services can be used to help people with serious and life-threatening medical conditions. They can provide a range of urgent healthcare and transport services, including acute care of patients who require help without a previous medical appointment. The crews within the Ambulance service include a range of medical staff, e.g., emergency care assistants and paramedics (131). Ambulance services often use a range of vehicles to respond to an emergency in the quickest possible time (132).

In our case study, we assume that initial medical emergency calls are dealt with by ambulance service operators (133). The call centre records the call and

captures the following information: name of patient, contact telephone number, address of patient, destination address and brief details of the patient's condition. This information is then transmitted to the ambulance region nearest to the patient for an ambulance to be dispatched. The regional ambulance centre receives the request and sends a vehicle and a crew to the patient, having selected the most appropriate service level: Immediate (i.e. life-threatening) and Urgent (i.e. serious but not life-threatening) (134). Next, the system specifies the location of the patient on a map and monitors the vehicle's progress towards the destination via online messages transmitted by each vehicle every 13 seconds (with the potential for the use of radio communication as a backup measure). The ambulance crew confirms that the vehicle is on its way. At the same time, the ambulance crew requests that the patient record be retrieved through the EHR service.

4.2.2 EHR Services

EHR services play a very important role in modern healthcare. Their main function is to capture, store and supply patients' records, including personal and medical information (e.g. data related to treatments and medication). There are a number of definitions for EHR. The US Institute of Standards and Technology defines an EHR *a* "*longitudinal collection of patient-centric health care information available across providers, care settings, and time. It is a central component of an integrated health information system*" (135). One of the benefits of EHR is to enable healthcare providers to share, store and update patient information in order to capture care history (136).

In our case study, the risks associated with EHR are mainly related to retrieving and updating patient information. This service can help users access the system and retrieve patient's EHR and update their records. The risks relate to conditions such as lack of information, incorrect information or late arrival of information.

4.2.3 Childbirth Services

Our case study is concerned with the transportation of pregnant women by the Ambulance Service. Every pregnancy is different and there is a wide variation in the length of labour (137). For first-time mothers, labour often takes

between 10 and 20 hours. For some, however, it lasts much longer, while for others it may be over much sooner. Labour generally progresses more quickly for women who already have children.

In our case study, there are risks associated with the transport of pregnant women in urgent cases. The principal risks are concerned with timing. In the first place, the fact that the emergency services have been contacted suggests that the delivery time is very close and there is a medical emergency. The ambulance station schedules a vehicle to go and collect the patient and take her to the hospital. The crew retrieves the patient information from EHR service. They examine the patient, check the maternal and foetal condition and transport the patient to the labour ward.

4.3 Service Hazard Analysis (SHA)

In this section, we introduce and describe SHA as the main safety analysis method for examining services in SOA and generating the necessary safety requirements. We also show how SHA has been applied to our healthcare case study.

4.3.1 Method

Safety analysis processes are centred on identifying, analysing and managing hazards. For SOA, identifying the hazards associated with the services is essential for defining the service safety requirements, which should influence and potentially drive the architectural design. In this section, the identification and classification of hazardous service failures is carried out by applying SHA to a high-level representation of the SOA in SoaML. Table 10 shows a tabular representation of the output of the analysis. The goal of SHA is to identify and clarify each failure mode associated with a service based on the defined deviations.

For each failure mode, SHA also identifies the potential effects of failure and determines the severity of these effects. The last two columns in Table 10 provide the final output of the SHA which is the starting point to generate and provide recommendations for the Service Safety Requirements (SSRs) to

mitigate the potential failures, and to allocate them to services, tasks and participants.

Service	**Failure Mode** **a) not provided** **b) provided when not required** **c) provided incorrectly**	**Effect**	**Severity**	**SSR**	**Mitigation**
Service name	*The specific failure modes associated with the service based on the defined deviations*	*The specific effects associated with the failure mode*	*The severity of each service failure mode effect*	*Definition of service safety requirements*	*Suitable recommendations for service safety requirements*

Table 10: SHA Table Template

SHA consists of six steps:

(1) Identify a service;

(2) Identify the service failure modes;

(3) Determine the safety effects of each service failure mode;

(4) Determine the safety severity and classification of each service failure mode;

(5) Provide service safety requirements;

(6) Identify potential mitigation measures for meeting the service safety requirements.

Step 1 - Identify a service. High-level services are captured in SoaML ServicesArchitecture models. The SoaML ServicesArchitecture model describes both Participants and Services and their interactions. A SoaML ServicesArchitecture model describes how Participants collaborate by providing and consuming Services to achieve goals. These collaborations can be captured in service Contracts. In this context, a service Contract is defined by means of an SLA, which describes a mutual agreement between two or more parties (138). These agreements are captured in the form of SoaML ServiceInterfaces between Participants.

Step 2 - Identify the service failure modes. For each service (identified from the SoaML ServicesArchitecture), the analyst considers the effects of three broad hypothetical failure modes:

- Service not provided when required;
- Service provided when not required;
- Incorrect service.

The service can be considered as a potential source of hazards in the SOA system and for every failure mode it is necessary to consider more than one interpretation (72). For instance, the first failure mode 'service not provided' is easily interpreted for responsive services (such as for requesting ambulances or allocating drivers to ambulances). From the perspective of on-going or regular services (such as monitoring the condition of patients or tracking ambulance locations), the 'service not provided when required' failure mode requires a more specific interpretation, e.g. failing to monitor the condition of patients every 3 minutes or failing to update ambulance locations every 5 seconds.

The failure mode 'service provided when not required' deals with uncommanded requests, e.g. requesting an ambulance when not required. From the perspective of on-going or regular services (e.g. maintaining available call centres), this failure mode might not be applicable or might be hard to interpret.

The third failure mode (incorrect service) covers a diverse set of behaviour. Possible explanations include wrong timing (i.e. too late or too early), incompleteness, asymmetry or undesirable side effects.

Eventually, these failure modes should be defined with enough detail in order to enable the generation of the service safety requirements and potential mitigation measures (e.g. by providing enough contextual information to make a specific interpretation).

Step 3 - Determine the safety effects of each service failure mode. The adverse consequences of the failure modes should be determined, taking different context-relevant factors into consideration. In terms of determining

the effects of service failures, a useful feature of the SoaML ServicesArchitecture models is that they include a representation of service interaction and the *Participants* that use these services, which makes it easier for the analyst to trace the effects of a particular failure mode. The analyst has to consider the effects of the failure modes on the system and on its environment. The analyst should consult people with operational experience to assist in classifying the failure mode effects.

In the case of the system level SHA, the effects of failure modes may be observed within the system level, or the analyst may have to consider combined effects of systems with their environment in order to determine the effect of the system failure mode.

Step 4 - Determine the safety severity and classification of each service failure mode. We determine the severity and classification of the identified failure modes by analysing accident and incident data, reviewing regulatory guidance material, using previous design experience, and consulting with system developers and operators.

The process of analysing and classifying the failure modes should be careful documented, and supporting materials (analyses, studies, tests, etc.) used should be preserved to ensure traceability for future reference. The safety classification should be defined (e.g. 'Catastrophic', 'Severe-Major/Hazardous', 'Major', 'Minor' and 'No Safety Effect'), typically based on Hazard/Risk Matrices (HRM) defined in the system's domain standards and regulations (139).

Step 5 -Provide Service Safety Requirements (SSRs). Based on the identified failure modes and their classification, recommendations should be generated. Preferably, these recommendations should take the form of SSRs, where the rigor/integrity with which the requirements need be met should be proportionate to the severity of the failures (i.e. higher degrees of severity require more stringent integrity requirements and more rigorous processes) (140).

Step 6 - Identify potential mitigation measures for meeting the SSRs. In order to help influence the design process, mitigation measures should be

identified that have the potential for meeting the SSRs identified in the previous step. For example, in order to ensure that hazardous behaviour is detected and controlled, the system might be designed to include active monitoring and cross-checking of the state of a service. Mitigations should build on best practice in safety and dependability design, e.g. existing work on tactics for fault tolerant services (120).

4.3.2 Healthcare Case study: SHA

In this section, we show how we applied the SHA method to the healthcare case study that we introduced briefly in Section 4.2. The aim of the case study is to illustrate the use of the method, understand how the SSRs can be developed in a systematic and traceable manner and provide a preliminary evaluation.

4.3.2.1 System Overview

This description of the system and all of the other information presented about the system and its design, including all design diagrams, are based on (128). The system architecture model in Figure 1 shows the structural architecture of the system. It basically covers the services used (i.e. produced and consumed) from the point at which a phone call is made to request an ambulance (for a pregnant woman) to the point at which the patient is admitted to a hospital (labour ward). Another set of models was created to capture subsequent stages in the patient's treatment e.g. foetal monitoring, first and second stages of labour, caesarean section and postnatal care. These models and documented in Appendix A for the sake of completeness, but are not analysed further in this thesis.

In Figure 1, we provide a design of the system which focuses on the interactions between services and Participants. Specifically, the scenario focuses on the Ambulance System architecture, and identifies five Participants connected by ten SoaML *Collaboration Uses*, i.e. services. These Collaboration Uses are defined in terms of service contracts, which are: Request Ambulance, Dispatch Ambulance, Update Status, Request Record, Provide Patient Record, Update Health Record, Retrieve Health Record, Pick up Patient, Examine Patient and Drop off Patient. In this section, we show how SHA has been

applied to two of the services: Dispatch Ambulance and Retrieve Health Record. The analysis of additional services is documented in Appendices B and C.

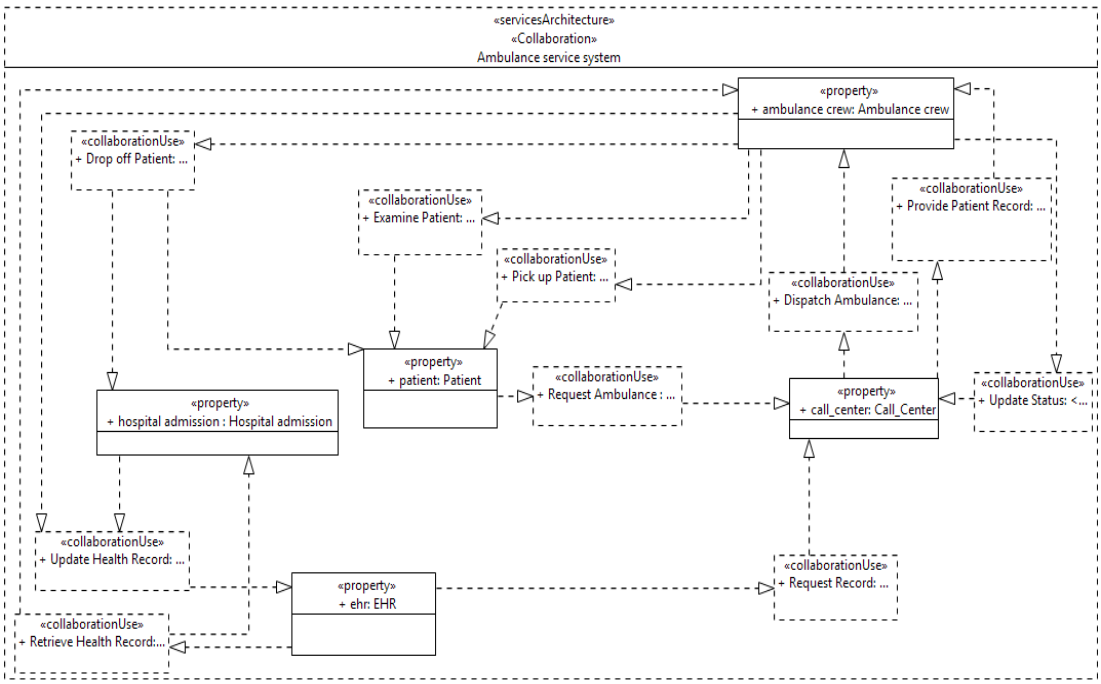


Figure 22: SoaML Service Architectures Model for SOA Healthcare

The Ambulance System architecture extract presented in Figure **23** shows how the Dispatch Ambulance Collaboration Use is defined to provide the service. Two Participants, *Call_centre* and *Ambulance crew,* have been specified, connected by the Dispatch Ambulance service contract. These two Participants represent the roles that are connected to this service.

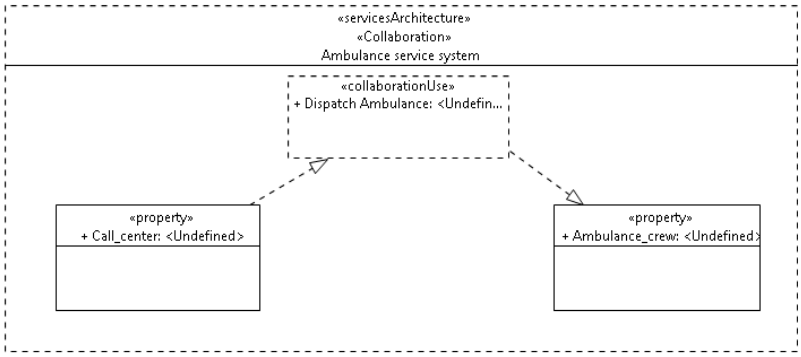


Figure 23: SoaML ServicesArchitecture Model (Dispatch Ambulance)

The description of the Dispatch Ambulance service is provided by means of a Service Level Agreement (SLA). The SLA is defined using the template, shown in Table 11, which has been adapted from (141).

Attribute	Description	Source
Name	The name of the SLA artefact, defined during SoaML service modelling	SoaML Service Contract Architecture
Provider	A person, department or organization that provides the service	SoaML Participants Architecture
Consumer	A person, department or organisation that uses the service	SoaML Participants Architecture
Start time	The start time of the SLA validity period	SoaML ServicesArchitecture
Due time	The end time of the SLA validity period	SoaML ServicesArchitecture
Duration	The length of time that the service should last for	SoaML ServicesArchitecture
(Pre-condition)	The condition that must be fulfilled before the service can start	SoaML Service Contract Architecture
(Post-condition)	The condition that must be fulfilled after the service is completed	SoaML Service Contract Architecture
Contribution to Safety	Required to complete an incident report	SHA

Table 11: SLA Template

Table 12 presents the SLA for the Dispatch Ambulance service. For instance, the Participants, Call_centre and Ambulance crew, represent the consumer and provider for the service.

Attribute	Description	Source
Name	Dispatches Ambulance	SoaML Service Contract Architecture for the ambulance system
Provider	The Ambulance crew department that provides the service	SoaML Participants Architecture for the ambulance system
Consumer	The Call_centre that uses the service	SoaML Participants Architecture for the ambulance system
Start time	Requesting an ambulance by the Call_centre (e.g. 11:00:00 AM)	SoaML ServicesArchitecture for the ambulance system
Due time	Dispatching of the ambulance (e.g. 11:00:30 AM)	SoaML ServicesArchitecture for the ambulance system
Duration	30 seconds	SoaML ServicesArchitecture for the ambulance system

(Pre-condition)	Call_centre has received the ambulance request information including the patient's name, contact telephone number, address of the patient, destination address and brief details of the patient's condition.	SoaML Service Contract Architecture for the ambulance system
(Post-condition)	The nearest available ambulance has been dispatched and the Call_centre has received an expected arrival time	SoaML Service Contract Architecture for the ambulance system
Contribution to Safety	Overall severity is major (see SHA Table 4)	SHA

Table 12: Dispatch Ambulance SLA

Table 13 shows an extract from the SHA analysis for the Dispatch Ambulance service. The analysis establishes the potential safety criticality of the Dispatch Ambulance service, based on the severity of the worst credible effects of the identified failure modes. The 'SSR' column details the stringent safety requirements which are allocated to the service as a result of the analysis, and the 'Mitigation' column indicates design and operational measures which are proposed to satisfy these requirements.

Service	**Failure Mode**	**Effect**	**Severity**	**SSR**	**Mitigation**
Dispatch Ambulance Context: *the patient has given birth before the ambulance arrives, and is actively bleeding following the birth*	Ambulance not dispatched	Death	Major	The failure mode 'ambulance not dispatched' shall be prevented or mitigated	Active monitoring and cross checking between requested and dispatched ambulances
	Ambulance dispatched when not required	N/A No direct safety effects but waste for critical services	N/A	None.	None.
	Ambulance dispatched later than intended	Severe morbidity (hypovolaemia, renal failure, cardiac arrest, disseminated intravascular coagulopathy…)	Major	The failure mode 'ambulance dispatched later than intended' shall be prevented or mitigated	Active monitoring of timing targets and strategies for recovery from timing-related failures
	Ambulance dispatched to the wrong address	Severe morbidity (hypovolaemia, renal failure, cardiac arrest, disseminated intravascular	Major	The failure mode 'ambulance dispatched to the wrong address' shall be prevented or mitigated	Early address cross-checking and confirmation between requested and dispatched ambulances

		coagulopathy…)			

Table 13: Dispatch Ambulance SHA

The first failure mode, 'ambulance not dispatched', is straightforward as this service is responsive in nature, i.e. a demand for an ambulance vehicle provided by an ambulance crew. The worst credible effect here as estimated by the medical expert, consulted in the process of this case study, is death and the severity is major as defined in the Health and Social Care Information Centre standard used for Health IT safety assurance in the NHS (142).

The defined SSR specifies the need to eliminate or mitigate this failure mode given its potential severity. The mitigation here, i.e. active monitoring and crosschecking, is considered appropriate, considering the identified potential effects of the failure (i.e. patient death) and the classification (Major).

Another failure mode that may occur is, 'ambulance dispatched when not required'. In this case there are no safety effects within the scope of the case study scenario (potentially the fact that an ambulance is unnecessarily occupied on this call means that it is unavailable for an emergency elsewhere).

A third potential hazardous failure mode is 'ambulance dispatched later than intended', which could arise from a timing-related failure and will result in delays to the patient's medical treatment. The potential effects, as identified by our clinical expert, are severe morbidities such as hypovolaemia, renal failure, cardiac arrest and disseminated intravascular coagulopathy. The severity is classified as major and the mitigation is active monitoring of timing targets and strategies for recovery from timing-related failures.

The fourth failure mode is 'ambulance dispatched to the wrong address', which may lead to delays in treatment. The effects identified and the severity classification are similar to those in the previous step but the mitigation is early address cross-checking and confirmation between requested and dispatched ambulances.

Table 14 and Table 15 show the SLA and the SHA results for the 'Retrieve Health Record' service. For this service, the two Participants are the EHR and Ambulance crew and the duration of the service is 20 seconds.

Attribute	Description	Source
Name	Retrieve Health Record	SoaML Service Contract Architecture for the ambulance system
Provider	The EHR provides the service	SoaML Participants Architecture for the ambulance system
Consumer	The ambulance crew uses the service	SoaML Participants Architecture for the ambulance system
Start time	The point at which a request for information is sent by the ambulance crew (e.g. 11:00:00 PM)	SoaML ServicesArchitecture for the ambulance system
Due time	The point at which the requested information is received by the ambulance crew (e.g. 11:00:20 PM)	SoaML ServicesArchitecture for the ambulance system
Duration	20 seconds	SoaML ServicesArchitecture for the ambulance system
(Pre-condition)	The system contains the correct patient details	SoaML Service Contract Architecture for the ambulance system
(Post-condition)	Correct health records for the patient have been received by the ambulance crew	SoaML Service Contract Architecture for the ambulance system
Contribution to safety	Overall severity is considerable (see SHA table 6)	SHA

Table 14: SLA for the Retrieve Health Record Service

Although the potential severity of failures of the Retrieve Health Record service is considerable, the worst credible consequence is not patient death, as it is assumed that the ambulance crew can recover from the failure modes associated with the Retrieve Health Record service. Also, in order to mitigate the inability to retrieve health records through the electronic system, the service relies on other communication means, e.g. radio communication.

Services	Failure mode	Effect	Severity	SSRs	Mitigation
Retrieve Health Record	EHR not retrieved	Delay in treatment and increased complication	Considerable	The failure mode 'EHR not retrieved' shall be prevented or mitigated.	Use of redundancy through fitting an additional flow to provide patient information from another source

	EHR retrieved when not required	N/A No direct effect on safety but waste for critical services	N/A	None	None
	EHR retrieved later than intended	Delay in treatment and increased complication	Considerable	The failure mode 'EHR retrieved later than intended' shall be prevented or mitigated.	Active monitoring of timing targets and strategies for recovery from timing-related failures
	Incorrect record retrieved	Incorrect treatments and medicines provided	Considerable	The failure mode 'inaccurate records retrieved' shall be prevented or mitigated.	Active monitoring and cross-checking of the information retrieved

Table 15: Retrieve Health Record SHA

4.4 Service Failure Analysis (SFA)

In this section, we introduce and describe SFA as the safety analysis method for examining the flow of the Tasks that implement the SOA behavioural process and for generating the safety requirements relating to task flows. More specifically, the objective of SFA is to examine the detailed flow between the service tasks and identify how failures, specifically interaction failures, can contribute to service hazards. We also show how SFA is applied to our healthcare case study. SFA extends SHARD (Section 2.3.1.3.) with three main aspects:

- It provides a traceable relationship between the causes of the failures and existing SOA fault taxonomies. This is discussed in Section 4.4.2.
- It provides an explicit means for defining derived service safety requirements as explained in Section 4.4.1. (Step 5).
- Finally, it offers a way for capturing tactics and design decisions for potentially meeting these derived safety requirements in the context of the SOA design. This is discussed is more detail in Section 4.4.1. (Step 6).

4.4.1 Method

SFA is centred on defining the deviations that can be exhibited in service tasks and understanding the causes and safety effects of those deviations. Based on these causes and the severity of the effects, SFA generates a set of safety requirements and recommends design patterns for mitigating the task deviations and therefore influences the detailed architectural design of the system. In SFA, BPMN models are used for representing the low-level design of the SOA. In particular, these models show the interactions of the system behaviour in the form of information flow between service tasks.

Based on the BPMN models, SFA uses a set of guidewords to consider potential deviations from the intended behaviour, i.e. failure modes, in the flows between tasks. The guidewords help ensure coverage of the potential failure modes in the context of the detailed design of the behavioural aspects of the SOA. When examining the causes of the deviations, SFA uses the SOA fault taxonomy developed by (143). This fault taxonomy is well structured and categorises service faults based on the SOA lifecycle phase in which they are generated, e.g. specific faults related to service execution and composition.

SFA results are represented in a tabular format as shown in Table 16. The columns should include ID (Flow), Guide word, Deviations, Possible Causes, Effects, Derived Service Safety Requirements (DSSRs) and Mitigations.

ID Flow	**Guide Word**	**Deviation Failure Mode**	**Possible Causes**	**Effects**	**DSSRs**	**Mitigation**
The flow link	*These describe the potential deviations from the intended behaviour in the flows between tasks*	*The specific flow failure modes associated with the task, based on the fault taxonomy*	*The potential causes of each flow failure mode of the specific tasks associated with the flow*	*The specific effects associated with the task, based on the fault taxonomy*	*Definition of DSSRs*	*Recommendations for design patterns and tactics for meeting the DSSRs*

Table 16: SFA Table Template

SFA consists of six steps:

(1) Identify a flow between two tasks;
(2) Identify flow failure modes;
(3) Determine the potential causes of each flow failure mode;
(4) Determine the potential effects of each flow failure mode;
(5) Define DSSRs; and
(6) Identify potential mitigation measures for meeting the DSSRs.

Step 1 -Identify a flow between two tasks. We use a behavioural model of the SOA as represented in BPMN (68) to identify the flows. BPMN provides the capability to communicate and represent internal procedures, based on Processes, in a graphical and structured notation. Typically, these Processes represent workflows of connected Tasks (i.e. atomic *Activities*), grouped into *Swimlanes* (i.e. a container for organising *Activities*). Further, each Participant in the model is represented as a Swimlane in BPMN. This step in SFA involves the selection of a link between two BPMN Tasks that captures a flow in terms of inputs, outputs, data, sequence and timing.

Step 2 -Identify flow failure modes. This step uses a number of guidewords, based on the SHARD guidewords, to determine the ways in which the selected flow can deviate from its intended usage. The failures derived from the use of each of these guide words need to be presented in the context of the detailed SOA design. The generic guidewords are taken from (72), but require some re-interpretation for use in the context of SOA:

- Omission: no delivery through the flow between the tasks (no communication);
- Commission: delivery through the flow between the tasks when not required (unintended communication);
- Early: delivery through the flow between the tasks earlier than intended (early communication);
- Late: delivery through the flow between the tasks later than intended (late communication);

- Value: wrong delivery of information through the flow between the tasks.

Step 3 - Determine the potential causes of each flow failure mode. Failures in flows can be caused by a variety of technical, human and organisational events or conditions, or by combinations of such events or conditions. For technical causes in particular, we use the SOA fault taxonomy developed by (56). This fault taxonomy is well structured and categorises faults based on the SOA lifecycle phase in which they can emerge: (1) publishing, (2) discovery, (3) composition, (4) binding and (5) execution. The advantage of using these fault types is that they are SOA-specific. However, these fault types should be used as prompts or hints for safety analysts rather than as an exhaustive list of all possible SOA faults.

Step 4 - Determine the potential effects of each flow failure mode. The potential effects associated with the failure types should be recorded and examined in terms of the contribution that they can make to the service hazards, identified in the SHA, or to a new hazard (i.e. hazards missed during SHA).

Step 5 - Provide DSSRs. Based on the identified failure modes and their causes, recommendations should be generated. Preferably, these recommendations should be in the form of DSSRs, where the rigor/integrity with which the requirements need be met should be proportionate to the failures (i.e. higher degrees of severity require more stringent integrity requirements and more rigorous processes) (140). Where a failure mode contributes to one or more hazards, one or more safety requirements should be defined to address the failure mode.

Step 6 - Identify potential mitigation measures for meeting the DSSRs. Design recommendations should be made for addressing the failure modes which have been identified during the analysis. These recommendations could be based on existing safety tactics in the software architecture literature (e.g. (50) and (58)). Importantly, in our approach, we select design approaches including detection of, containment of and recovery from identified classes of

failure. More specifically, for SFA, we adopt the following categories of design tactics (120):

1. *Redundancy - Invokes one or more copy of the same mechanism;*
2. *Diversity - Invokes one or more copy of a particular mechanism that performs the same function;*
3. *Monitoring - Checks the system continuously to identify failures and sends alerts;*
4. *Diagnosis - Identifies the source of failure;*
5. *Masking - Hides the effects of a failure;*
6. *Containment - confines a failure to stop its propagation;*
7. *Recovery - Erases failure and restores normal operation.*

4.4.2 SOA Fault Taxonomy used in SFA

In this section, we introduce the SOA fault taxonomy that we use to support the choice of mitigation measures in SFA. This SOA fault taxonomy is based on the types of fault defined in (144) and (145). As shown in Figure **24**, these faults are categorised based on the lifecycle phase in which they can occur: Publishing, Discovery, Composition, Binding, and Execution. These faults provide the analyst with a generic list of potential causes and should be treated as guidance rather than as a complete list of all possible causes.

4.4.2.1 Publishing Faults

During the publishing phase, faults can occur due to issues related to service task description or deployment.

Faults relating to the service task description occur when problems arise when defining the characteristics of the service. Either the service is not completely described, or the defined service task. These types of faults may lead to problems either during the discovery phase or during the execution phase. Faults might also occur when the description of the file is wrong. Format faults can occur when the format of a file is incorrect. For example, an XML file might not be well formed due to missing tags.

Faults can also occur when the deployment of a service is incorrect (service task deployment faults). For example, a service task deployment fault might

occur when the service is deployed without the required resources being available.

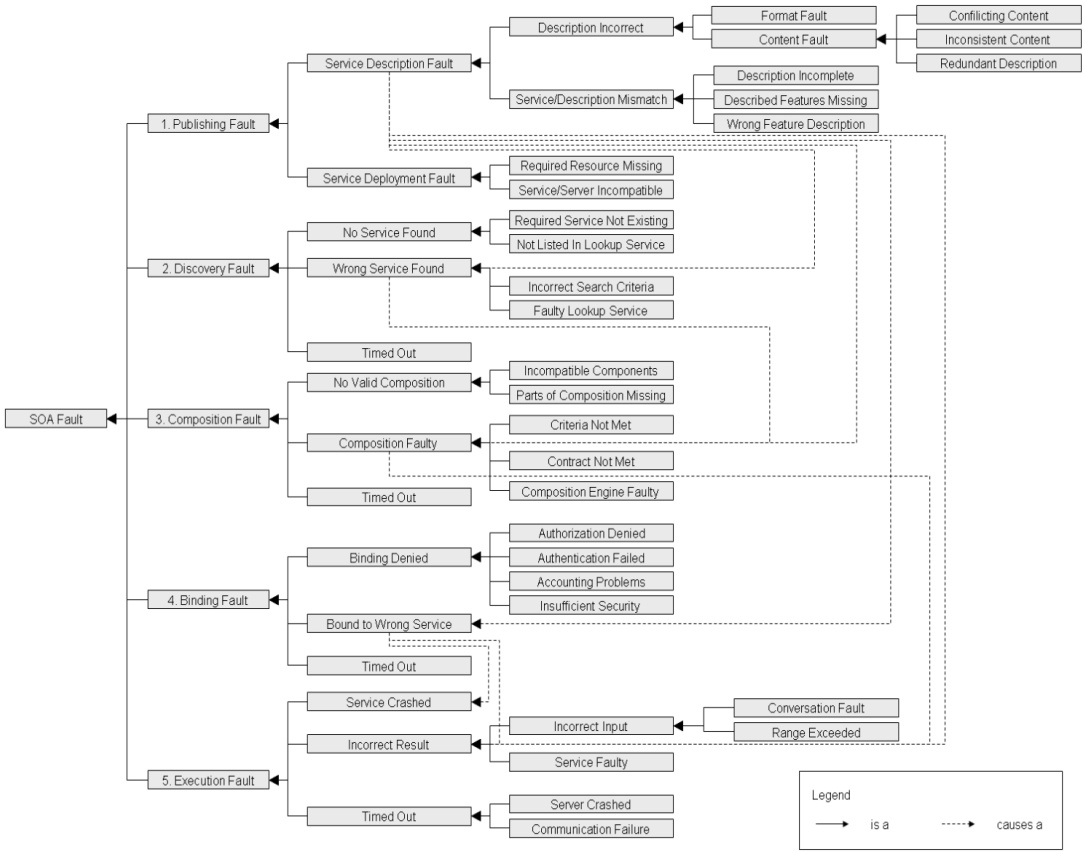


Figure 24: Taxonomy of SOA-Specific Faults (146)

4.4.2.2 Discovery Faults

In the discovery phase, three possible types of fault may occur, namely:

- Service not found;
- Wrong service found; and
- Time out.

The first of these is the most likely to occur, i.e. the required service task does not exist. This can be due to incorrect search criteria being used, or to the task not being listed in the lookup service. The second fault type, wrong service found, is difficult to detect in this phase but is more likely to be detected in the execution phase.

The time out fault is a common fault in the discovery, composition, binding, and execution phases. Time out is typically caused by the server crashing as a result of hardware, software or communication errors.

4.4.2.3 Composition Faults

During the composition phase, faults can occur because of invalid composition due to missing parts or timed out attempts. Figure 25 provides an example of compatible service task failure in which a 'no valid composition' fault occurs due either to incompatible components or to missing components.

More importantly for services, composition faults can result from an inability to meet contract conditions: pre-conditions, post-conditions and invariants. In our work, these conditions are explicitly defined in SLAs (initially defined at the service level in the SoaML models and considered in the SHA).

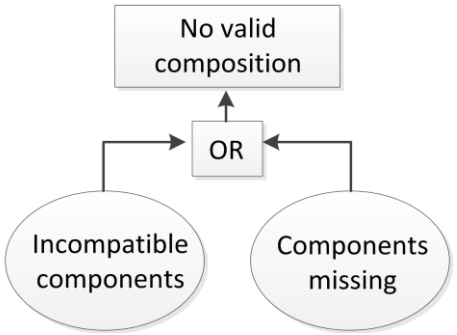


Figure 25: 'No Valid Composition' Failure Mode

4.4.2.4 Binding Faults

In this phase, the service task consumer and provider negotiate conditions needed to execute the task. Types of faults which can arise during the binding phase include:

- Binding denied;
- Bound to the wrong service task; and
- Time out.

The 'binding denied' fault can occur when the authorisation for use has not been granted by the authorisation component. The 'bound to wrong service' task is tightly linked to service description faults that occur during the publishing phase.

4.4.2.5 Execution Faults

Execution faults occur when the outcome of a service task does not match the result expected by the participant. Relevant failure types here include:

- Incorrect result;
- Service task crashed; and
- Time out.

During execution, the service is run and an incorrect result might occur if the wrong service task has been selected. The 'incorrect result' fault might arise either due to a software fault (service faulty) or to an incorrect input. Incorrect inputs may be caused by conversion faults or by the input range being exceeded.

In the SFA method, the SOA fault taxonomy discussed above (Section 4.4.2) is principally used for identifying the causes of the flow failure modes identified in Step 2 of the SFA. These causes are linked to a specific type of faults identified in the SOA fault taxonomy. The flow failure modes are assessed in terms of the contributions that they make to the service hazard failures (i.e. as identified in SHA).

4.4.3 Healthcare Case study: SFA

In this section, we show how we applied the SFA method to the healthcare case study that we introduced in Section 4.2. The objective of the case study is to illustrate the use of the method and to provide a preliminary evaluation.

4.4.3.1 Representing Tasks and Information Flows in the SOA Design

The modelling presented in this section builds on the results of the SoaML modelling in Section 4.3.2.1. We have already specified the Service Contracts for the healthcare case study, using SoaML, and have refined these into Service Interfaces in the system architecture design in Figure 27. The Service Interfaces offer a number of Operations that specify the interactions between the Participants of the architecture. In our approach, and in order to integrate the SoaML models and BPMN models of the SOA design as shown in Figure 26, we link SoaML Operations with BPMN Tasks, enabling traceability between the

different models at different abstraction levels (i.e. structural services and behavioural service tasks and interactions).

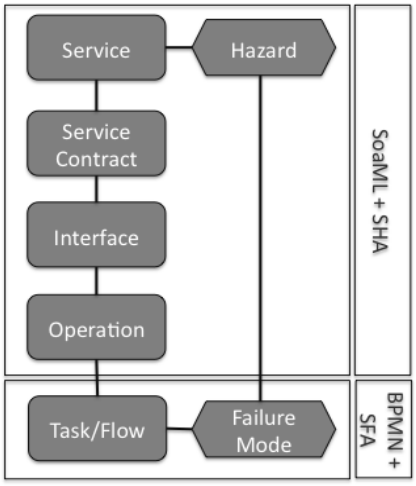


Figure 26: Link between SoaML and BPMN Models

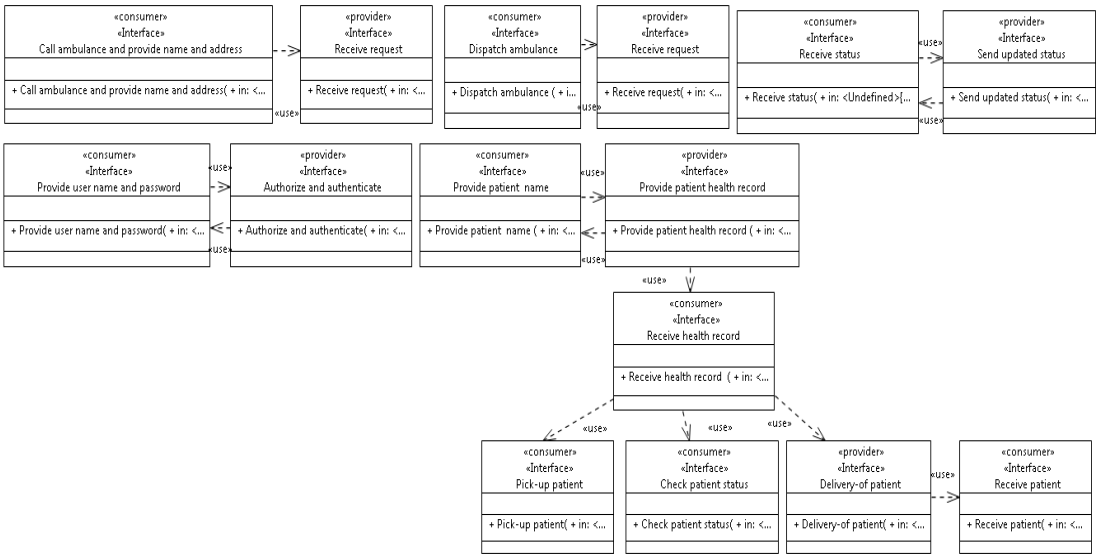


Figure 27: SoaML Service Interfaces Model with Operations

For example, the Retrieve Health Record service can be divided into a number of operations within the SoaML interface model and has been linked to the BPMN Tasks, which are as follows:

- Provide user name and password;
- Authorise and authenticate;
- Provide patient name; and
- Provide patient health record.

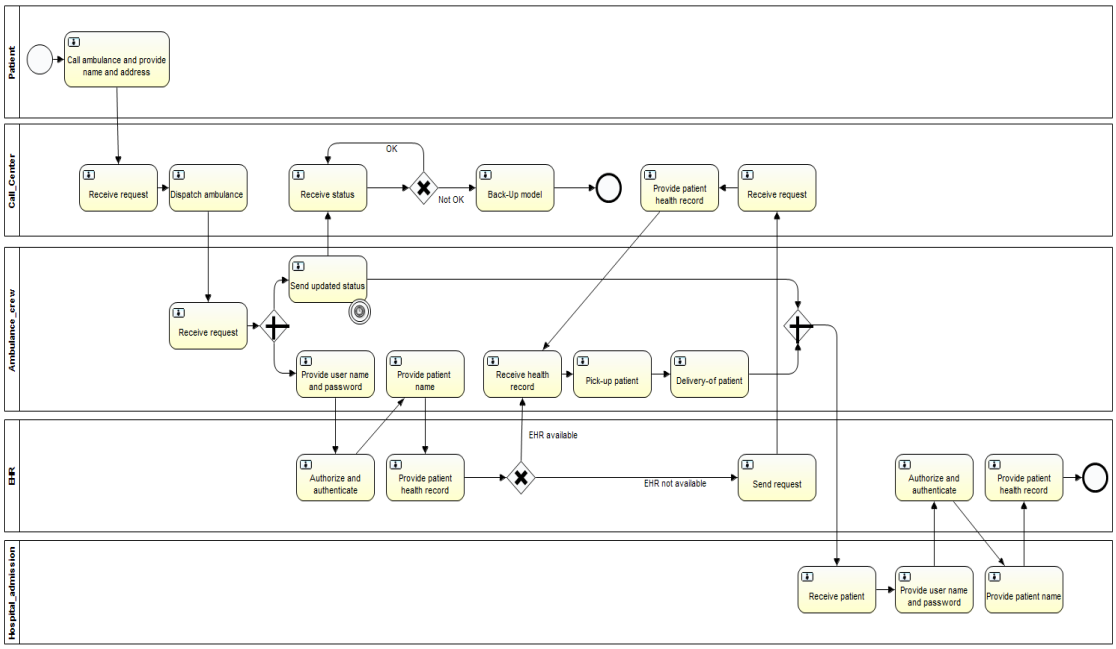


Figure 28: Tasks in BPMN Model

Figure 28 presents a more detailed BPMN model of the SOA design, with more emphasis on the SOA behavioural processes. BPMN Tasks in these processes are mapped into the Operations in the Interfaces provided by each SoaML Service, while BPMN Swimlanes are mapped onto the SoaML Participants. SFA was applied to every flow between Tasks in the BPMN model. An example is shown in Table **17**, which shows the results of the analysis of the penultimate flow in the BPMN model, from 'Provide patient name' to 'Provide patient health record' (in the context of admission to the labour ward). The example shows how a lack of patient information from the EHR, and more seriously the provision of incorrect information, can potentially lead to adverse patient complications.

ID (Flow)	**Guide Word**	**Deviation**	**Possible Causes**	**Effects**	**DSSRs**	**Mitigation**
Flow between 'Provide patient name' and 'Provide patient health record'	Omission	No record available	Authentication failure or request timed out (server crashed)	Incomplete history and background Anaphylaxis-unknown allergy status	The failure mode 'no record available' shall be prevented or mitigated.	Use of redundancy in data sources for health records

	Commission	Patient record provided when not requested	False messages Description incorrect or mismatch Execution fault (Incorrect result)	No direct safety effects	The failure mode 'patient record provided when not requested' shall be prevented or mitigated.	Monitoring of uncommanded requests
	Early	N/A	N/A	N/A	N/A	N/A
	Late	Record arrived later than intended	Failure in scheduling Time out related to clock time with ripple effects on all processes	Overloading due to dealing with backlog requests	The failure mode 'record arrived later than intended' shall be prevented or mitigated.	Monitoring of time delays and recovery by giving more priority for delayed requests
	Value	Incorrect record retrieved	Incorrect input or conversion fault Corruption message execution fault incorrect result or service crash	Anaphylaxis-unknown allergy status	The failure mode 'incorrect record retrieved' shall be prevented or mitigated.	Data entry cross-checking, online monitoring and fault containment

Table 17: SFA for the Flow between 'Provide Patient Name' to 'Provide Patient Health Record'

The first identified failure mode (omission) is 'no record available when requested'. Possible causes, as generated from the SOA fault taxonomy, are authentication failures and request time out (i.e. due to a server crash). The effects might be incomplete history and background, which could result in the patient's suffering anaphylaxis due to unknown allergy status. The mitigation for this failure mode is the provision of redundancy in data sources for health records. Another failure mode that may occur is revealed through the application of the guideword 'commission': a record could be sent even though it had not been requested. Possible causes for this are false messages, incorrect data description, mismatch or incorrect result. Here there are no direct safety effects and the mitigation suggested is monitoring the arrival and reporting of uncommanded requests.

Application of the guideword 'late' reveals a possible failure mode in which the patient's record is received later than intended. Possible causes include failures in the scheduling system or the clock time on all processes (i.e. timed out processes). The effect that the system is likely to become overloaded with a backlog of delayed requests. The mitigation suggested is monitoring of time delays and recovery of the system by giving delayed requests a higher priority. The guideword 'value' reveals a potential failure mode 'incorrect record retrieved' which can result in the patient's suffering anaphylaxis due to the medical staff having no data of a relevant allergy. The mitigation measures suggested for this failure mode are cross-checking of data entries, online monitoring and fault containment.

ID (Flow)	**Guide Word**	**Deviation**	**Possible Causes**	**Effects**	**DSSRs**	**Mitigation**
Flow between 'Dispatch ambulance and 'Receive request'	Omission	No request for ambulance received	Deployment fault (Request resource missing) Service/server incompatible Task crashed	Ambulance not dispatched	The failure mode 'no request for ambulance received' shall be prevented or mitigated.	Use of redundancy in data sources by Fit addition flow to the request
	Commission	Uncommanded request for ambulance received	False request, incorrect request Execution fault (Incorrect result)	No safety effect	N/A	N/A
	Early	N/A	N/A	N/A	N/A	N/A
	Late	Request arrived later than intended	Failure in scheduling/ synchronisation Time out related to clock time with effects on other tasks	Ambulance dispatched later than intended	The failure mode 'request arrived later than intended' shall be prevented or mitigated.	Monitoring for timing failure and developing recovery strategies
	Value	Incorrect request	Incorrect input or conversion fault execution fault,	Delay in the treatment	The failure mode 'incorrect request' shall be	Diversity via programming solutions to detect and reject

			incorrect result or servicer crash		prevented or mitigated.	improper values and fault containment

Table 18: SFA for Flow between 'Dispatch Ambulance' to 'Receive Request'

Table **18** shows another example of the use of SFA for examining the flow between the 'Dispatch Ambulance and 'Receive Request' tasks. An important issue to note here is that two of the effects highlighted by the analysis, 'ambulance not dispatched' and 'ambulance dispatched later than intended', are the direct causes of the service hazards identified in the SHA analysis in Table 13. This shows the continuity of the safety analysis between SHA, at the service level, and SFA, at the detailed design level.

4.5 Tool Support

This section describes the implementation of the tool support for the SOA modelling and safety analysis which has been described in this chapter, and presents the metamodels on which it is based. The tool support environment provides the following the capabilities:

- Create SoaML and BPMN models;
- Create and integrate SLAs into the SoaML models;
- Integrate BPMN and SoaML models;
- Embed SHA results into the SoaML models;
- Embed SFA results into the BPMN models.

These capabilities are intended to support the design and safety analysis of the SOA system at the high (i.e. services) and low (tasks) levels of the design. The following subsections illustrate the implementation and use of these capabilities.

4.5.1 SoaML to BPMN Link

In this thesis, we implemented most of the models in Eclipse (124) using the SoaML and Activiti BPM plug-ins (125) and (126). We have extended the SoaML plug-in to perform the traceability with the BPMN models. Figure 29 shows the implementation of the SoaML-BPMN link within the SoaML *ServicesArchitecture* model through the selection of an appropriate file for the

BPMN Model. Then, the specific notation element and the ID are chosen, in order to determine the precise Tasks within the BPMN model that are linked to the SoaML model.

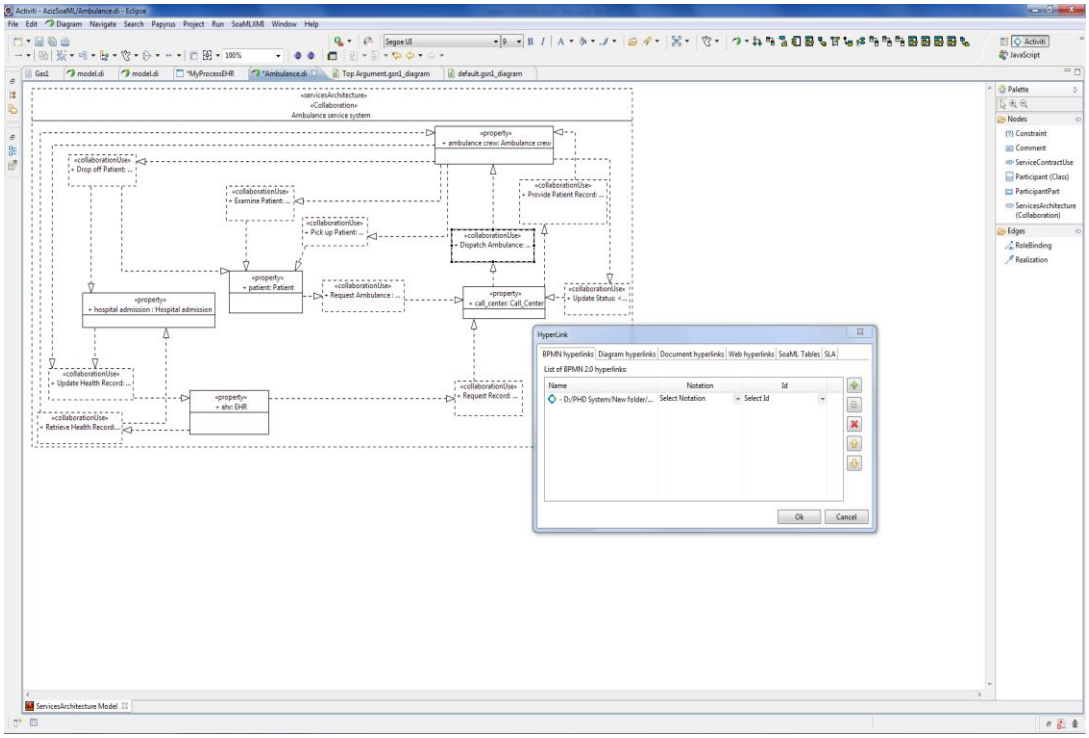


Figure 29: Link from SoaML ServicesArchitecture Model to BPMN Model

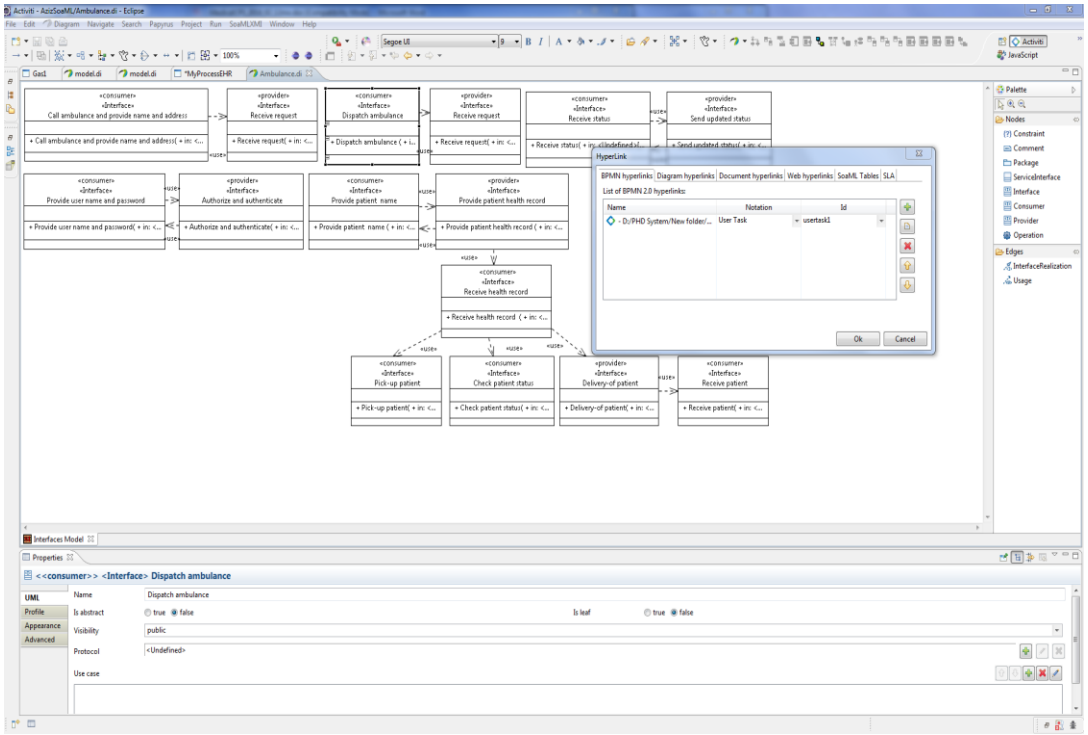


Figure 30: Link from SoaML Interfaces Model to BPMN Model

Once the traceability between the SoaML and BPMN models is established, each of the Operations within the SoaML Interfaces model can be linked to a specific Task in the BPMN model as shown in Figure 30.

4.5.2 BPMN to SoaML Link

Traceability between the SoaML and BPMN models can also be established. Figure 31 shows the traceability between lower-level design models (BPMN) and high-level service models (SoaML ServicesArchitecture model). The link is established first within the BPMN model by selecting a SoaML File. Then, the specific model (e.g. ServicesArchitecture model), the Stereotype for the model (e.g. Participant or CollaborationUse or ServicesArchitecture) and the Name of the chosen element are linked to the precise service in the SoaML models. The example in Figure 10 shows the link between the 'Dispatch Ambulance' Task in the BPMN model and the 'Dispatch Ambulance' service in the SoaML *ServicesArchitecture* model.

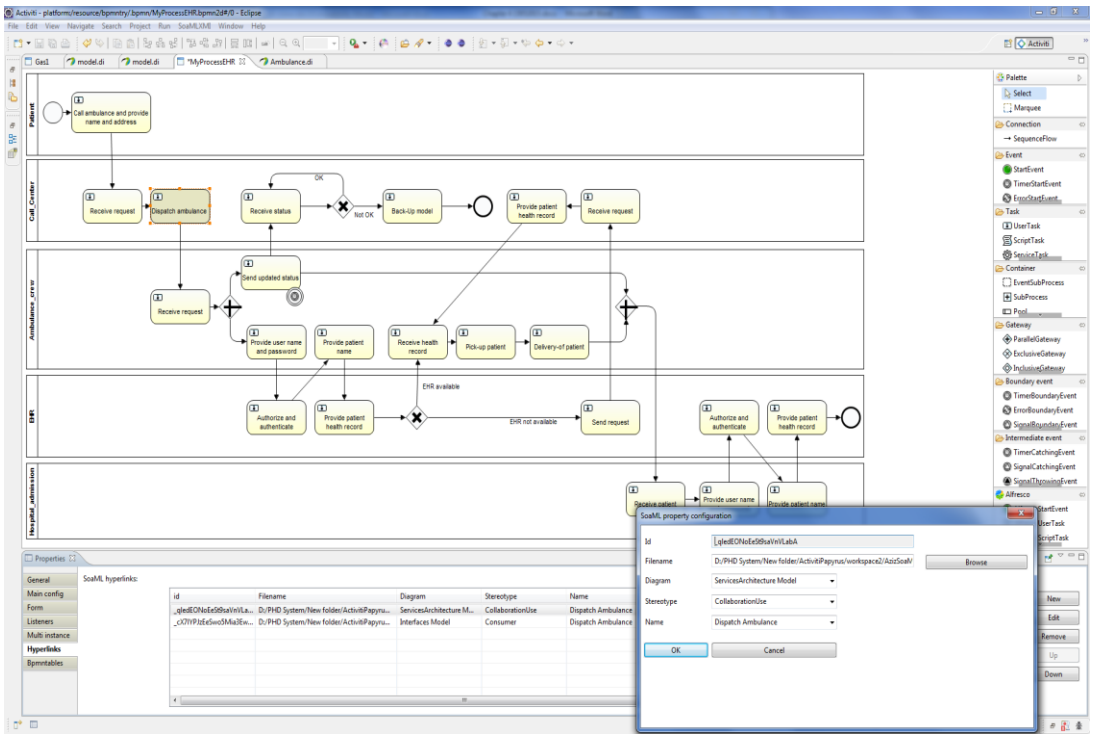


Figure 31: Link from BPMN to SoaML ServicesArchitecture Model

As before, once the traceability between the SoaML and BPMN models is established, Tasks described in the BPMN model can be linked with Operations captured in the SoaML Interface model.

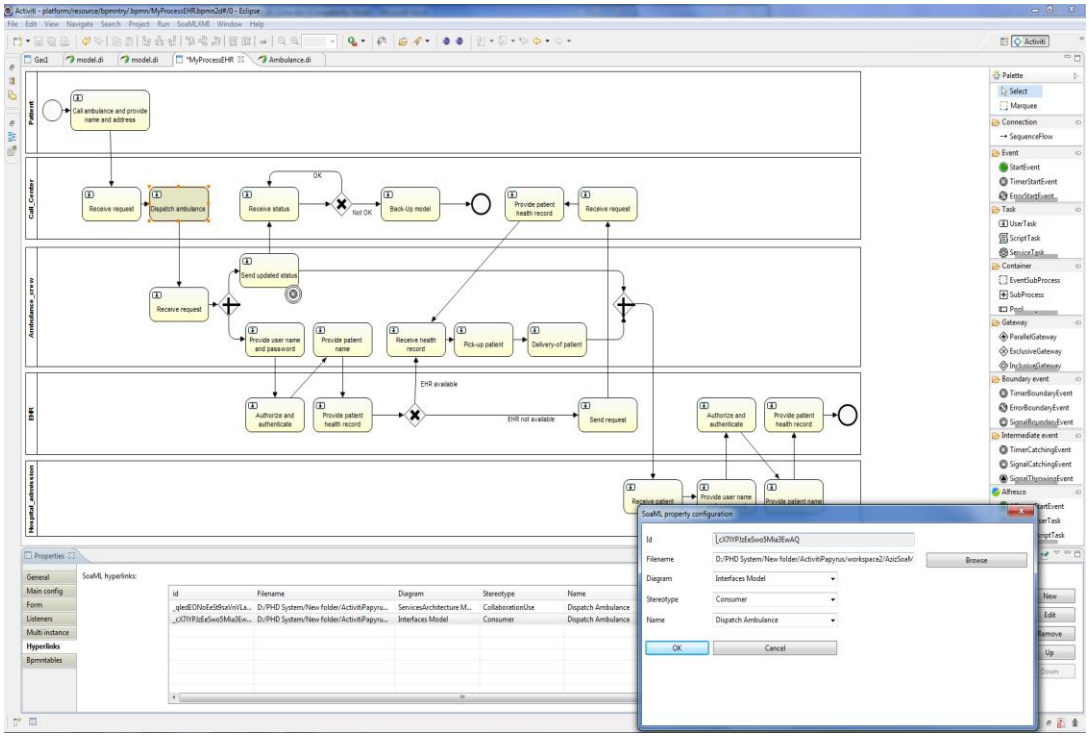


Figure 32: Link from BPMN model to SoaML Interface Model

For example, in Figure 32, the 'Dispatch Ambulance' Task is linked with the 'Dispatch Ambulance' operation in the SoaML Interface model.

4.5.3 SLA Implementation

In this section, SLA is represented as part of the SoaML service contract. The example presented in Figure 33 shows the creation of an SLA description and the recording of the pre- and post-conditions for the service and the required information for SLA.

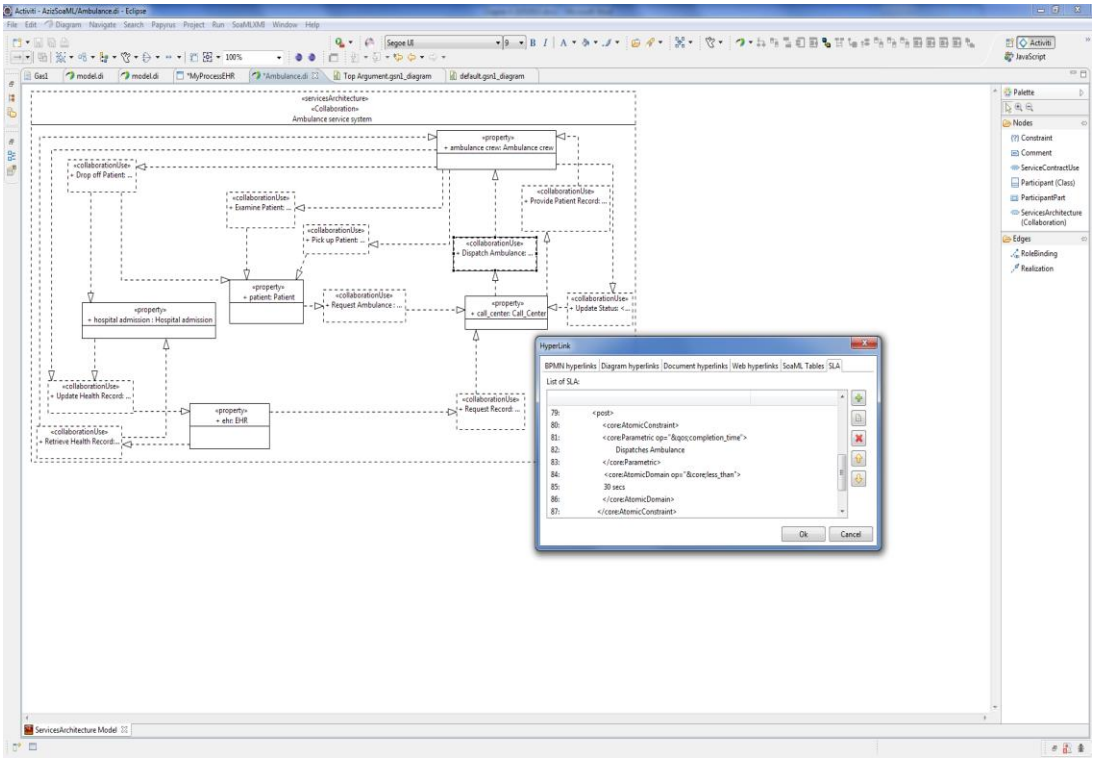


Figure 33: Representing XML schema for SLA

4.5.4 SHA Implementation

This section describes the tool support for SHA as integrated into the SoaML models. Figure 34 shows the SHA user interface for populating the service safety data. When the user clicks on a service, the SHA results relating to that service are displayed and can also be edited. More specifically, in the example, the tool displays the data corresponding to the Dispatch Ambulance service failure mode 'ambulance dispatched later than intended', its effects (hypovolaemia, renal failure, cardiac arrest, disseminated intravascular coagulopathy…), the severity of the effects (major), proposed mitigations (active monitoring of timing targets and strategies for recovery from timing-related failures) and SSRs (e.g. "ambulance dispatched later than intended shall be prevented or mitigated").

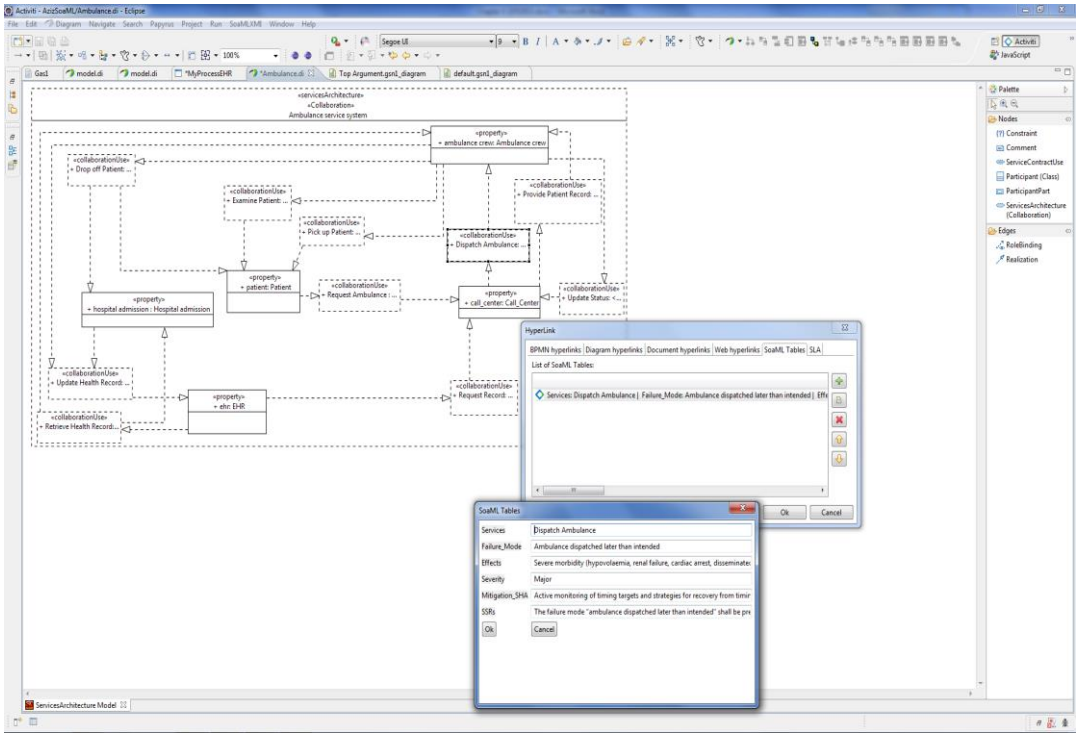


Figure 34: SHA implementation

4.5.5 SFA Implementation

This section describes the tool support for SFA as integrated into the BPMN models. Figure 35 shows the SFA user interface for populating the service safety data. When the user clicks on a flow link, the SFA results relating to the selected flow are displayed and can also be edited. More specifically, in the example, the tool shows the data corresponding to the flow between 'Dispatch ambulance' and 'Receive request' for the guideword 'Omission'. The deviation identified is 'no request for ambulance received', the possible causes are 'Deployment fault (Request resource missing), Task/server incompatible and Task crashed', the effect is 'Ambulance not dispatched', the DSSR is "the failure mode 'no request for ambulance received' shall be prevented or mitigated" and the proposed mitigation is "use of redundancy in data sources by fitting an additional flow".

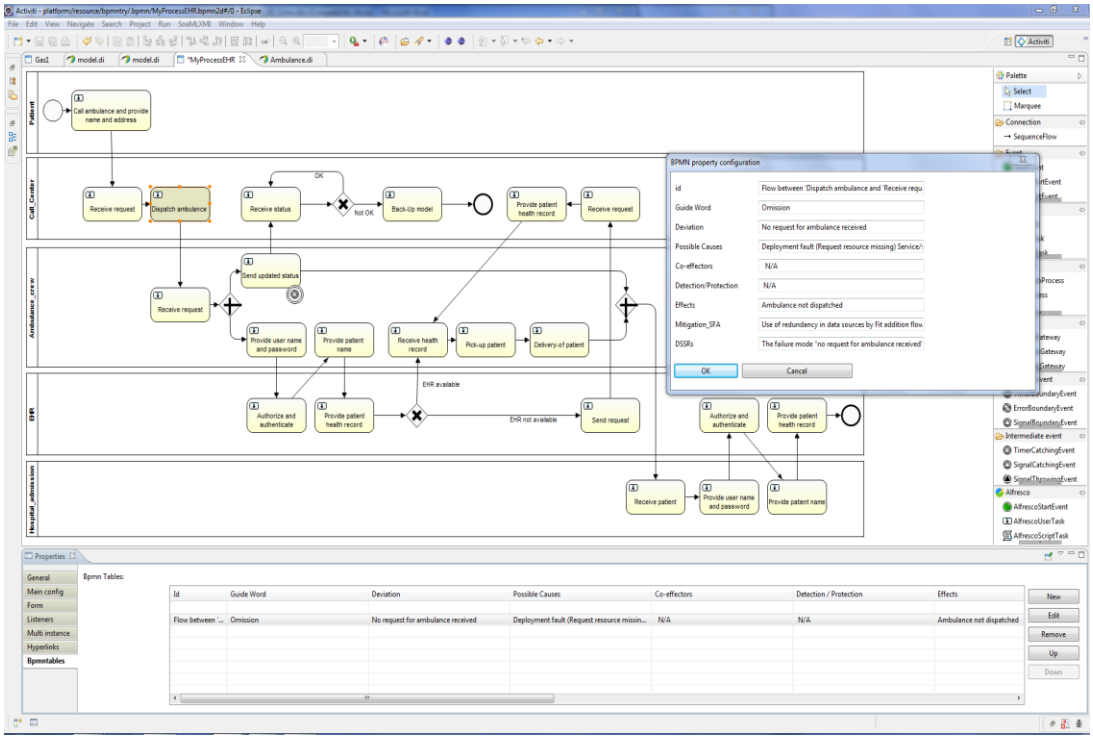


Figure 35: SFA implementation

4.5.6 Model-Based Support

This section describes the models and metamodels used to support the tool implementation. Figure 36 shows the extension of part of the SoaML metamodel by adding *BPMN hyperLinks* to link the SoaML models with the BPMN models, *SoaML table* to capture the SHA data, and *SLA* to capture SLA pre- and post-conditions.

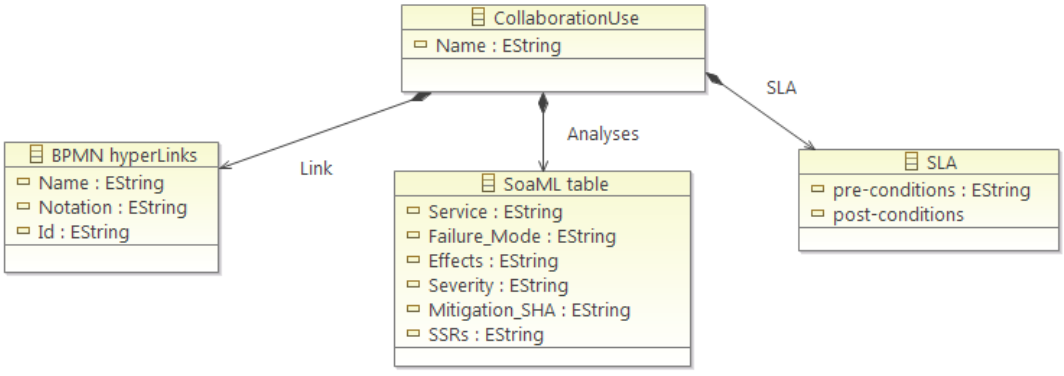


Figure 36: Extended SoaML ServicesArchitecture Metamodel

Figure 37 another extension of the SoaML metamdel to create and support the link between SoaML participants and BPMN tasks (*BPMN hyperlinks*).

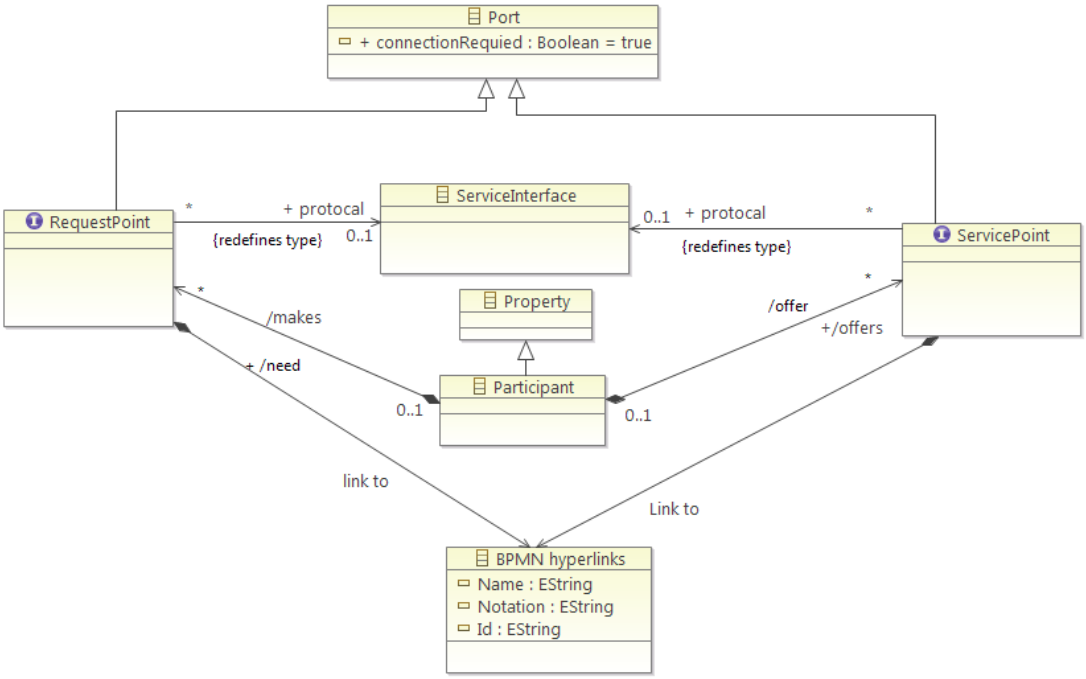


Figure 37: Extended SoaML ServiceInterfaces Metamodel

Figure 38 shows the extended BPMN metamodel, with the addition of the *Hyperlinks* (to link to the SoaML metamodel) and *Bpmntable* (to capture the SFA results).

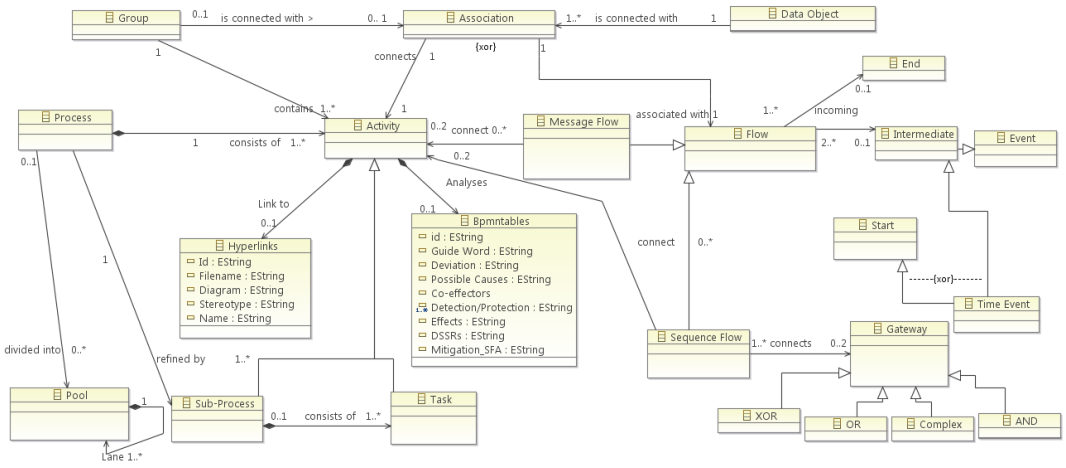


Figure 38: Extended BPMN Metamodel

4.6 Summary

In this chapter, we have presented an SOA safety analysis process that is integrated with the modelling of the SOA design. The safety analysis captures and examines the hazards and failure behaviours relating to services and tasks

according to the way in which these services and tasks are defined in the SOA context. This in turn forms the basis for defining the safety requirements and for recommendations for satisfaction of the requirements, which should influence and drive the design process. Producing these requirements can lead to an integrated and verifiable approach to assuring the safety of the SOA system. We have also introduced a detailed case study from the healthcare domain. The case study provides an illustration of the SOA safety analysis techniques used (SHA and SFA) and how they are applied and allows for evaluation of the techniques defined. Finally, we discussed the implementation of the tool support that provides capabilities for SOA modelling, safety analysis and traceability management. The information generated from the SOA safety analysis process forms the basis for assuring the system by means of safety cases as described in the next chapter.

Chapter 5

5. SOA Safety Cases

5.1 Introduction

In Chapter 2, we discussed the background to safety, various safety-related issues and certification processes. Safety certification was described as an important step in assuring the safety of a system, and the safety case was introduced as a means of recording the arguments and evidence necessary for assurance and – in some domains – for certification. The role of the safety case is to present a clear and defensible argument that the system is acceptably safe to operate within a specific application and to indicate the evidence on which the argument depends. Modular GSN (99) has been developed as a notation to represent safety arguments for modular systems. In this chapter, modular GSN will be the basis for our structuring and representation of SOA safety cases.

Chapter 3 introduced an overview of the SOA safety assurance framework that we have developed to assess how an SOA system can meet its safety requirements and can be assured by providing an explicit safety case. Chapter 4 described the detailed safety analysis methods that we adapted for SOA, namely SHA and SFA.

In this chapter, we develop an approach to the justification of safety-critical service-based systems, using modular GSN to represent the compositional structure of the safety case. This chapter explores the relationship between the service architecture and the safety case architecture. We identify the similarities and differences between both architectures. For example, the

structure of the SOA system should ideally correspond to the safety case structure to help ensure that they meet architectural attributes such as maintainability (i.e. by incorporating maintainability in the SOA system design, we should be able to achieve corresponding maintainability in the safety case, reducing the impact of system change on assurance). In particular, contracts which are defined to enforce the relationships between the safety argument modules will be based on the SLA between the actual services.

In this chapter, the information used to structure and the evidence used to support the safety case will be based on the results of the modelling SoaML for services and BPMN for tasks and flows as well as on the safety analysis outcomes generated from SHA and SFA.

Finally, the general structure of the safety arguments for SOA will be captured in the form of GSN patterns. These patterns will be organised in modules that correspond to the overall SOA structure, e.g. services, SLAs and participants, and explicitly linked to the results of the SOA modelling and safety analysis. The argument patterns are presented at a generic level, and can be instantiated to form argument structures relating to the assurance of specific SOA systems.

This chapter is organised as follows. In section 5.2, we introduce modular safety case development for SOA as a means for improving design and assurance traceability. Section 5.3 presents the elements of our SOA safety argument approach. Sections 5.4 and 5.5 describe the relationship between the SOA safety argument approach and the results of the SOA modelling and safety analysis described in the preceding chapters, and give practical guidance on how to create SOA safety arguments based on a pattern catalogue. Finally, Section 5.8 presents the instantiation of the approach based on our healthcare case study.

5.2 Modular Safety Case Development for SOA

In many safety-critical industries, it is regarded as best practice to construct a safety case in parallel with the system development, and to use the safety requirements to drive the design and development processes in such a way as to generate claims and evidence for the safety case. Because of the nature of

SOA systems, which may contain several interacting services, developed and operated by different service providers, establishing and justifying an acceptable level of confidence in the overall safety of SOA-based systems will often require integration of different safety arguments and evidence generated by different service owners or providers. One approach to representing this compositionality of the overall safety case for service-based systems is through modular GSN (97). Modular GSN supports the definition of the high-level safety case as a composition of different, yet mostly interrelated, argument modules, each of which addresses some aspect of the overall system. In order to improve flexibility and maintainability and support reconfigurability, some argument modules are connected using contracts (15). A contract is a special kind of GSN argument module which "*contains a definition of the relationships between two modules, defining how a claim in one supports the argument in the other*" (97), hence promoting loose coupling between argument modules and minimising the impact of change.

In order to improve traceability in the design and assurance processes, it is desirable for a safety case to have a clear correspondence with the design of the system. Essentially, an SOA is mainly a modular architecture, where each service represents a separate high-level module with defined intentions, interactions, inputs and outputs.

This modularity in the definition of services lends itself to the concept of modular safety cases, where there are explicit reply-guarantee relationships between the different safety argument modules (147).

A GSN argument module provides justifications relating to a specific aspect of the safety case (93). SOA-systems are modular in nature, i.e. individual services can be treated as individual modules (99). In our approach, each of the services in the overall system architecture will have a corresponding safety argument module and the overall safety case will address the composition of these safety argument modules. The same will apply for detailed design elements such as participants. Contracts between the safety argument modules will be based on the SLAs (Brown, Fenn, & Menon, 2010) that are defined between the actual services.

Services and participants are self-contained system elements that interact through predefined interfaces (e.g. those provided through SLAs). Service owners or producers often rely on other services when guaranteeing and providing their own services (122). Similarly, claims in certain argument modules can only be said to be substantiated (the guarantee clause) if claims or evidence are available in other argument modules that offer sufficient support (the rely clause).

5.3 Modular SOA Safety Argumentation Approach

This section briefly introduces the elements of our SOA safety argumentation approach, how it is applied and how it relates to the results of the SOA modelling and safety analysis described in Chapter 4.

5.3.1 Overall Structure

This section considers the overall organisation of the modular safety cases that we developed for SOA-based systems and discusses how the safety case argument relates to the structure of the SOA and the safety analysis. The purpose of the arguments is to justify the safety of the SOA design. This is done by arguing that the system safety requirements, which arise from the safety analysis, have been addressed through the selected design mitigations and provide for a sufficient level of safety.

Our SOA safety analysis follows a structured, though implicit, line of reasoning: an SOA-based system is acceptably safe because all credible service hazards have been identified and mitigated. This is supported by the safety analysis of services and their interactions provided by SHA which identifies relevant service hazards to be addressed in the argument. Furthermore, this line of reasoning is supported at a lower level through the analysis of the technical failures that could potentially contribute to the service hazards. This is supported by the safety analysis of tasks and flows provided by SFA.

We capture this reasoning through an argument pattern (123) to ensure consistency across the modular argument and allow the safety analyst to reuse the argument pattern as a basis for the assurance of different SOA-based systems (99). We have developed a catalogue of SOA safety argument patterns, which provides a framework in which an argument regarding the safety of

using an SOA system within a specific safety application can be developed. The argument patterns in the catalogue are defined separately, but are directly related to each other to form the overall argument structure. The SOA argument pattern catalogue includes rules for pattern composition and traceability to the architectural design (i.e. SoaML and BPMN models) and analysis (i.e. SHA and SFA).

The SOA safety argument pattern catalogue consists of templates for three individual arguments and one argument contract Figure 39. These patterns are structured to form overall hazard-directed argument for the SOA (Top Argument), which is supported by a number of Service Hazard Arguments, SLA Argument Contracts and Participant Arguments. These patterns can be composed together to form a complete argument regarding the safety of the SOA system design.

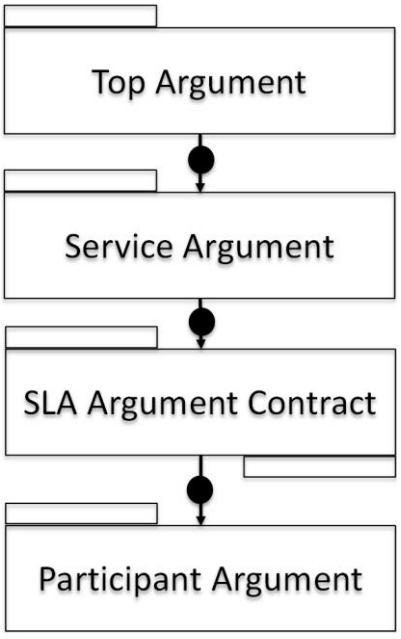


Figure 39: SOA Argument Patterns Catalogue

Within the pattern catalogue, we have provided means to represent the argument regarding the safety of the SOA system from the perspectives of both the Services and the Participants. The SOA safety argument pattern catalogue is explained in detail in Section 5.7. In the following subsections, we summarise each of the template safety argument modules in the SOA safety argument pattern catalogue Figure **39**. The SOA safety argument pattern catalogue follows a top-down chain of reasoning from the Top Argument module through to the

Participant Argument module. The instantiation of each pattern provides the necessary contextual information for the next pattern.

5.3.2 Top Argument

This module contains a hazard-directed argument, which covers the main high-level safety claims concerning adequate mitigation of the identified service hazardous failures. These claims are supported by arguments and evidence provided in the Service Arguments – safety argument modules defined to address the safety assurance of services that comprise the SOA.

5.3.3 Service Argument

Each Service Argument module contains mitigation arguments for the hazards posed by a given service and its interactions. These claims are supported by the SLA Argument Contract modules defined to address the safety assurance related to participants (producers and consumers of services).

5.3.4 SLA Argument Contract

The SLA Argument Contract modules include the detailed SLA Contract argument concerning the relationship between services and participants within the modular safety case. The contracts between safety argument modules are based on the SLA, which can help to identify the interdependencies that exist between services and participants and their corresponding safety case modules. These contracts are supported by post-conditions, relating to the services, and pre-conditions, relating to the participants, which specify relevant issues required from the supporting modules to provide safety assurance for the services.

5.3.5 Participant Argument

Participant Argument modules address assurance concerns associated with SOA participants. These modules are intended to provide sufficient support for the SLA argument contact modules. This includes providing the detailed arguments concerning the tasks relating to the relevant participant and interactions based on the pre-conditions allocated to the participant.

5.4 Relationship between SOA Modelling and Modular Safety Cases

The key modules in the design and safety case of an SOA are shown in Figure 40.

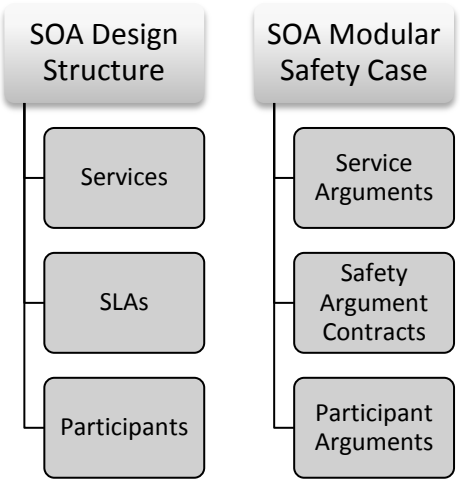


Figure 40: SOA and Modular Safety Case Correspondence

We force a traceable relationship between the high-level SOA design, in SoaML (*ServicesArchitectures*) and the overall SOA safety case, in modular GSN. For each service defined in the SOA, we create a Service Argument that addresses the hazards posed by the service (based on the results of the SHA analysis), i.e. the argument provides a justification for how the service hazards have been mitigated. In the same way, for each participant, we create a Participant Argument that justifies the contributions that the participant makes to service hazards (based on the analysis results of SFA). Moreover, participants interact through defined SLAs in order to provide and consume services. These interactions between participants can contribute to safety and therefore their assurance is addressed in explicit safety argument contracts (SLA Contract Arguments).

Before we present the definition of SOA safety arguments (Section 5.7), it is important to discuss the relationship between the SOA structure and the modular structure of the safety case argument. More specifically, this section explores the relationships between the modelling of the services, contracts and participants that comprise an SOA and the structure of the modular safety case argument for the system.

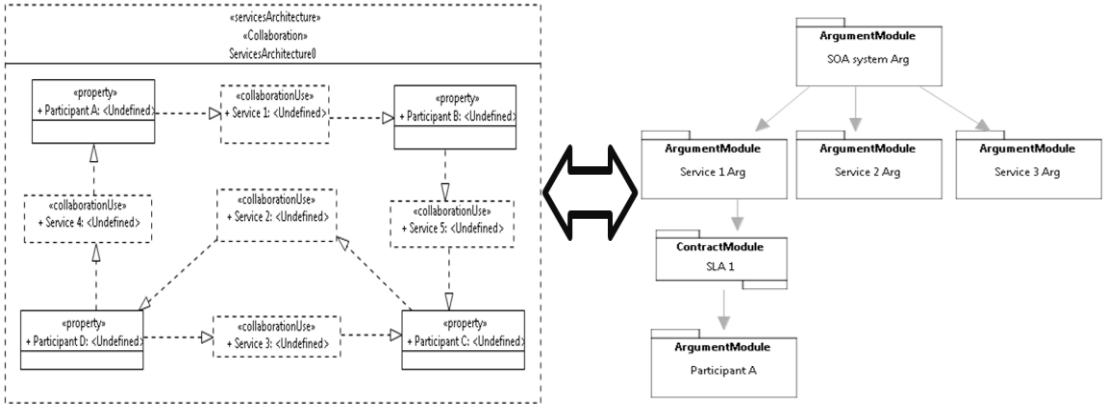


Figure 41 : Relation between SOA architecture and module safety cases

Figure 41 illustrates the potential relationship between the SOA organisation and the modular safety case structure outlined above. SoaML ServicesArchitectures models are used to describe the main components and communications within the system, the relationships between services and participants and how they work together. The modular safety case captures the structure of the service safety argument.

In Figure 41, each service, represented as a SoaML CollaborationUse, has a corresponding Service Argument module. Contracts between the argument modules are based on the SLA between the actual services as defined in the SoaML model (ServicesArchitectures). Each SLA will have a corresponding SLA Argument Contract module. Similarly, each participant will have its corresponding Participant Argument module and is linked to the BPMN pool that models the tasks and flows between tasks that define the behaviour of the participant.

5.5 Relationship between SOA Safety Analysis and Modular Safety Cases

In this section, the discussion focuses on how the nature of the SOA safety analysis (described in Chapter 4) relates to the overall structure of the SOA safety argument. Figure 42 illustrates the relationship between the SOA safety analyses, SHA and SFA, and the overall SOA safety case. In our argumentation approach, the main sources of reasoning and evidence are generated from the SHA and SFA results.

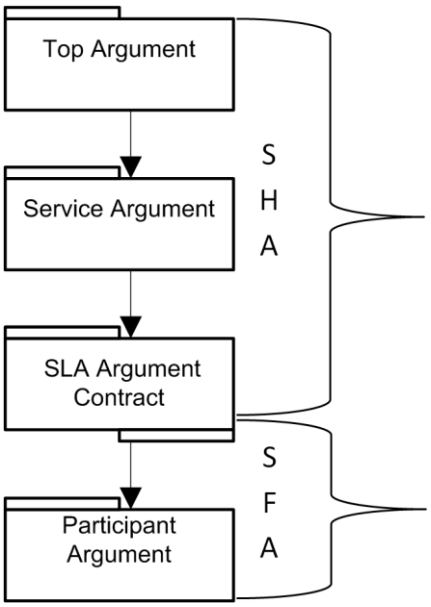


Figure 42: SOA safety analysis and structure of the SOA safety argument

5.5.1 Service Hazard Analysis (SHA)

SHA results play a significant role in defining the overall structure of the SOA safety argument patterns, specifically the high-level argument. For each service in the SOA which is related to safety, our approach requires the instantiation of an argument, based on the Service Argument pattern, in order to justify the claim that potential hazardous failures of the service are appropriately mitigated or managed. The argument approach is driven by the SHA results. These results support claims about the nature of the hazardous failures and how they are mitigated, based on a set of service safety requirements whose stringency depends on the criticality of the failures with regard to overall service safety. The claims made about the mitigation of the failures depend on how the mitigation measures are satisfied by the participants involved in provision and consumption of the service. These measures are modelled using SLAs in the SOA models. In our SOA safety case approach, we discuss the adequacy of the mitigation provided in a corresponding safety argument contract, where the claims relate to post-conditions based on the recommended mitigation measures generated from the SHA results.

5.5.2 Service Failure Analysis (SFA)

The pre-conditions of the SLAs, discussed in the previous section, are allocated to participants that are expected to implement the services in terms of specific

tasks and interactions. The aim of the Participant Argument pattern is to provide assurance that all the task flows have met the safety contract pre-conditions arising from SLA argument pattern. Therefore, the second half of the overall SOA safety argument pattern is influenced by the SFA results, which are organised according to deviations in the flows from the intended behaviour allocated to the participants involved in the provision of a service. This is captured in the Participant Argument pattern. This pattern captures the justification for the management of potential failures in the provision and consumption of tasks allocated to a participant. The definition of these failures is generated from the SFA analysis and the strategies chosen to manage their causes are justified in the context of different categories of failure modes defined in the SOA fault taxonomy, as discussed in Chapter 4. The pattern also shows that for each flow between participant tasks, the mitigations suggested by the DSSRs have been deployed and are sufficient to guarantee the required behaviour.

5.6 Managing SOA Safety Cases using GSN

In this section, we review the concept of safety case patterns and techniques which have been developed to allow them to be represented graphically. We also introduce our approach for SOA safety. The Goal Structuring Notation (GSN) has been used to present the structure and contents of our SOA safety argument pattern catalogue. The catalogue uses both the patterns and modular extensions of GSN (97).

5.6.1 Safety Case Patterns

The safety case pattern concept was originally developed by Kelly (123) based upon the work of the architect Christopher Alexander (36). Alexander's challenge was to try to discover a technique for town planning and structural engineering that could be reused to solve planning problems in multiple contexts. In his book, *The Timeless Way of Building*, Alexander shows how patterns can be used to abstract away from the details of specific buildings and environments, and can capture fundamental aspects of the design that can be applied elsewhere. Alexander asserts that:

"Each pattern describes a problem which occurs over and over again in our environment, and then describes the core of the solution to that problem, in such a way that you can use this solution a million times over, without ever doing it the same way twice." (36).

Kelly in (123) adopted Alexander's principles concerning the use of patterns for documenting recurring problems, solutions and contexts, and developed the concept of patterns for GSN safety arguments. Kelly defined a safety case pattern as "*a means of documenting and reusing successful safety argument structures*". (123)

The Goal Structuring Notation (GSN) was initially developed for the description of concrete safety arguments for specific systems and contexts. Kelly extended the original GSN technique with elements that provide the ability to represent an argument in a more generic format, i.e. as an abstract argument pattern, so that it can be applied to construct new safety arguments for different systems and contexts.

5.6.2 GSN Pattern Extension

GSN provides a means for capturing safety arguments and referencing the available evidence. GSN can be used to represent a specific safety argument for a given system or capability. Nevertheless, to be able to apply the notation in a more generalised way, GSN provides an extension for creating patterns and supporting abstraction, namely optionality, multiplicity extensions and entity abstractions. The pattern extension is illustrated in Figure 43 (adapted from (123) and (106)). The core elements of the GSN notation and methodology were introduced in Chapter 2. Here, we restrict our discussion to the extensions to support pattern definitions. These extensions rely on two types of abstraction (Figure 43):

- Structural Abstraction – supporting generalised n-ary, optional and alternative relationships between GSN elements (Multiplicity Extensions - Optionality Extension)
- Entity Abstraction – supporting generalisation/specialisation of GSN elements (Entity Abstraction Extensions).

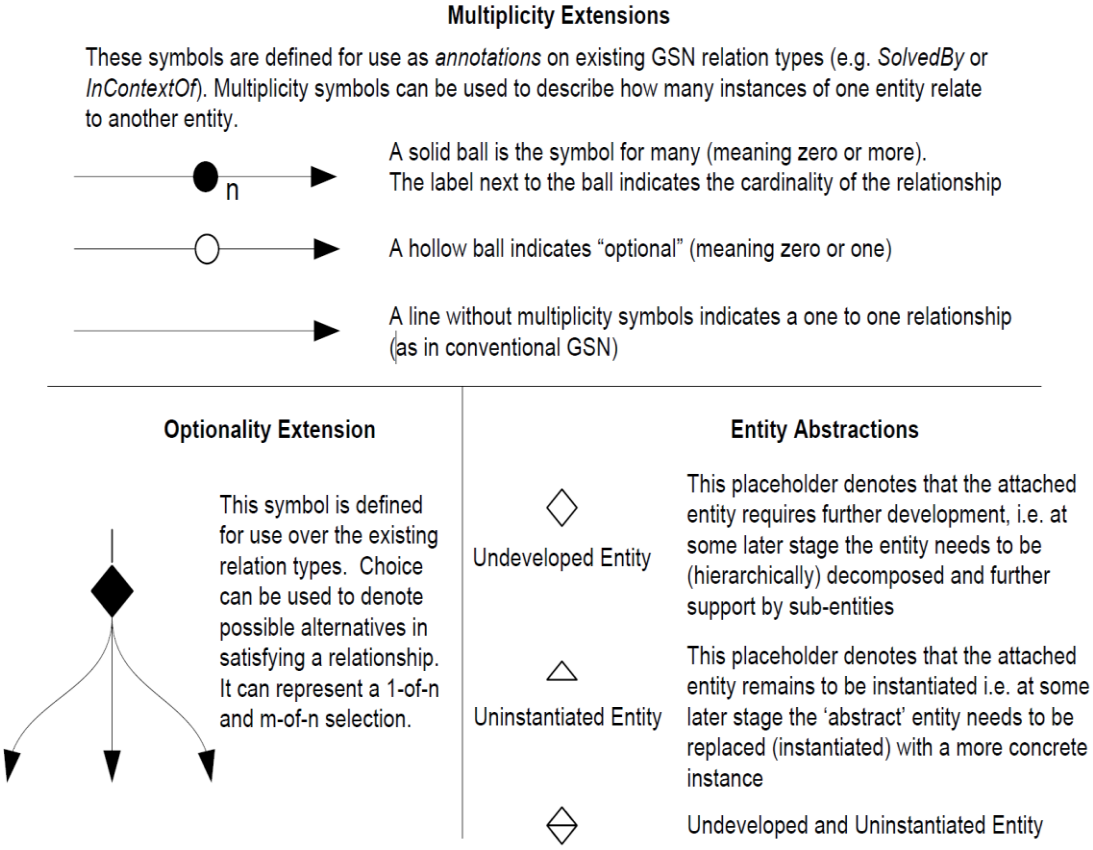


Figure 43: Optionality/multiplicity extensions and entity abstractions (106)

GSN has been also extended explicitly to allow support for the concepts of modular safety case construction (99). Figure 44 shows the key elements supporting modularity.

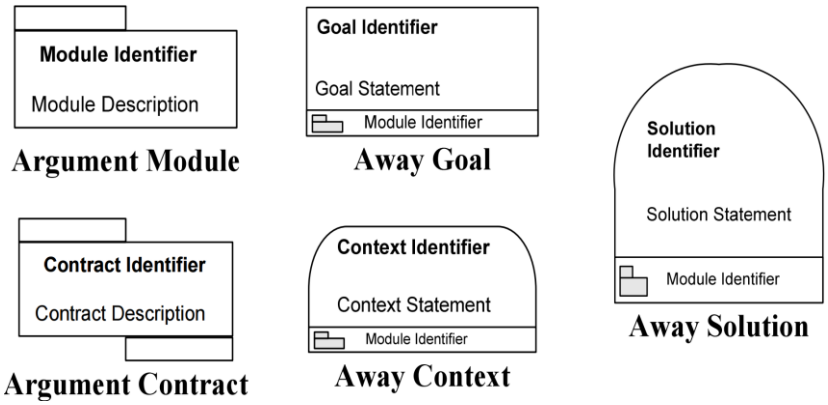


Figure 44: GSN elements introduced to handle modularity

The Argument Module is an explicit representation of a self-contained argument structure and is used a container for the elements used to capture that argument. The Away Goal element is used in order to be able to refer to

goals (claims) presented in one argument module which are satisfied by argument and/or evidence contained within other modules. Away Contexts and Away Solutions are used in order to make reference from the argument of one safety case module to context and evidence that exist within the boundary of another. Argument Contract symbols can be used to preserve the overall integrity of the modular safety case when the internal details of one or more argument modules are modified. The modular and pattern extensions of GSN are discussed in more detail in (97).

5.6.3 Documenting Safety Case Patterns in GSN

In this thesis, we use the template developed by (14) in order to document safety case patterns. The template requires the following categories of information (taken from Kelly, 1998):

Pattern Name

The pattern name should briefly convey the essence of the pattern. This should communicate the central argument being presented through the safety argument pattern. This will be the label by which people will identify this pattern.

Intent

This statement should answer the question "what the pattern is trying to achieve/do?"

Also Known As

This is used if there are other names by which the pattern could be described or recognised.

Motivation

This section gives a brief account of why the pattern was constructed. This usually includes scenarios that show a safety issue / process and how the elements of the GSN solve the problem.

The scenarios help understand the more abstract description of the pattern that follows.

Applicability (Necessary Context)

This section records under what circumstances the argument can and should be applied. Any assumptions and principles underlying the argument pattern are

identified in this section. For example, what are the circumstances in which the safety case pattern can be applied? What information is required (necessary inputs to the pattern) to be successful? How can you recognise circumstances in which the pattern can be deployed?

Structure

A graphical representation of the structure of the argument pattern is presented using GSN.

Participants

This section provides more description of the argument pattern structure, including a description of each of the elements of the safety case pattern (i.e. the goals, the contexts, the strategies and the solutions). The element descriptions make clear their function within the overall argument pattern. These elements should also state whether the element requires development or instantiation when the pattern is deployed.

Collaborations

This section describes how the different elements of the pattern work together to achieve the desired effect of the pattern. This section also explicitly identifies where links between elements exist that are not communicated by the argument structure.

Consequences

This section identifies what work remains after having applied the argument pattern.

Implementation

This section should communicate how the application of the pattern should be carried out, including any hints or techniques which would ease successful application of the pattern. It should also highlight possible problems (or pitfalls) in applying the pattern as well as common misinterpretations of the terms or concepts.

Example Applications

This section provides examples that illustrate the instantiation of the pattern.

Known Uses

This section describes known uses of the form of argument presented in the pattern.

Related Patterns

This section identifies any other safety case patterns that are related to the pattern.

5.7 SOA Safety Argument Pattern Catalogue

Figure 45 shows the overall organisation of the argument pattern catalogue we have defined for the justification of SOA safety. The following sections contain a detailed explanation and documentation of the argument patterns.

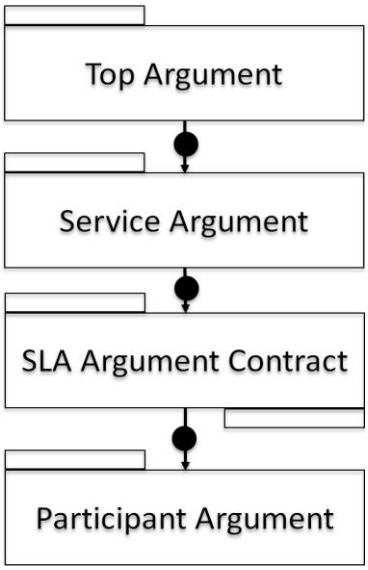


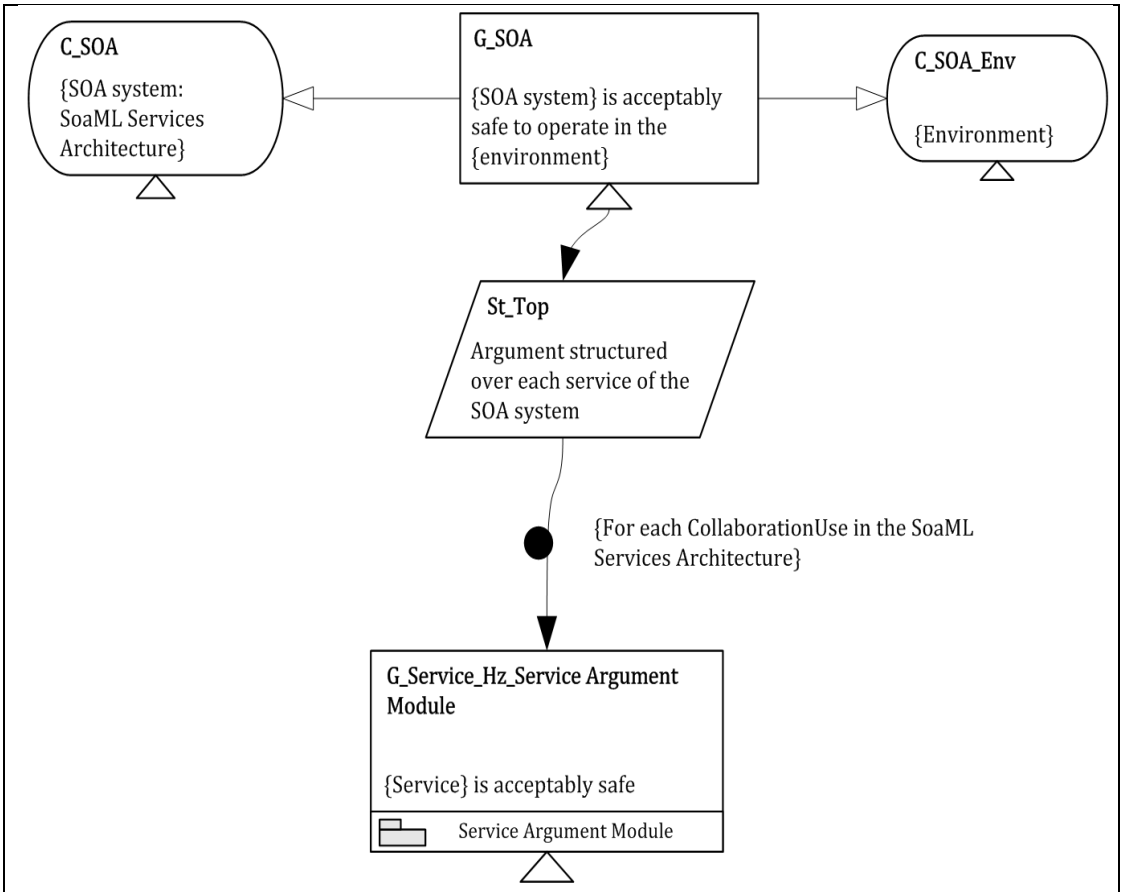
Figure 45: SOA Argument Patterns Catalogue

5.7.1 Top Argument

This pattern provides the top-level decomposition for the argument structured over the services. Table 19 shows the documentation of the pattern in GSN. The instantiation of this pattern starts with the *G_SOA* goal, which is the claim of overall safety for the SOA system. The structure of the pattern is driven by the architecture of the services within the system, starting from the SOA system and identifying how the services within the system can contribute to safety. The top-level claim is made in the context of the description of the SOA system and the environment in which it will be deployed. The strategy adopted is to partition the argument over all of the identified services, considering each service in turn. The context for this is based on the service specification in SoaML. For each service, the argument contains a claim that the service is acceptably safe. This is made in the context of the SOA design (this is taken

from the SoaML model). In order to show that any service hazards have been mitigated, an away goal is defined to argue separately that there are measures in place to address the safety of the service (this claim is substantiated in the Service Argument Module).

Top Argument			
Author	Abdulaziz Al-Humam		
Created	08/07/2014	**Last Modified**	04/07/2015
Intent	The intent of this pattern is to provide the high level and overarching argument to justify the safety of the SOA-based system, taking into account, explicitly, the services comprising the system.		
Also Known As	SOA Top Argument Pattern		
Motivation	SOAs are increasingly used within safety-critical applications. The safety of an SOA system must be justified, to which end a safety argument should be provided as part of the system safety case. This pattern was developed to provide a structured top-level argument for the safety of an SOA system used within safety-critical applications.		
Structure			



Participants	G_SOA	The overall objective of the argument is to provide sufficient support for the claim that the use of {SOA System} in the intended context is acceptably safe. {SOA System} is instantiated from the SoaML ServicesArchitecture. The link to this model is important to provide a specific and traceable description of the system that is the subject of the safety argument.
	C_SOA	This context provides a description of the SOA system, including all services within the system. This is

		provided by the SoaML ServicesArchitecture.
	C_SOA_Env	This considers the scope of the SOA system by defining the boundary of it, explaining the domain of the system. It also captures the assumptions about the behaviour of the environment that are made by the SOA system. Unless this behaviour can be guaranteed, the SOA's behaviour cannot be guaranteed. This is often stated in terms of requirements on the environment (e.g. in the form of user manuals).
	St_Top	This presents the strategy (the decomposition of the argument into claims about each service) adopted to support G_SOA.
	G_Service_Hz_Service Argument Module	This claim argues the safety of each service and is captured as an away goal. This claim references the top claim made and supported in a separate argument module relating to the relevant service: i.e. the Service Argument Module for that service.
Collaborations	• C_SOA and C_SOA_Env provide necessary context for the understanding of the system and its environment. • C_Top_Hz provides the link with the safety analysis,	

	SHA, used for the identification of the hazardous contribution of the SOA.
Applicability	This is a general pattern and, as such, has a wide applicability. This pattern is applicable wherever safety arguments need to be established for the use of services within a safety-critical application. However, it assumes that systematic SOA modelling and safety analysis have been performed, based on SoaML and SHA.
Consequences	After instantiating this pattern away goals will be instantiated in the Service Argument Modules. The Service Argument Module for SOA Safety Argument pattern can be used to decompose this goal.
Implementation	Top Argument Module should be fully developed and instantiated. Possible problems include lack of understanding of the environment or lack of knowledge of safety engineering concerns in the domain of the system (e.g. healthcare or aviation).
Examples	See the two case studies in this thesis.
Known Uses	See examples above.
Related Patterns:	Service argument Pattern.

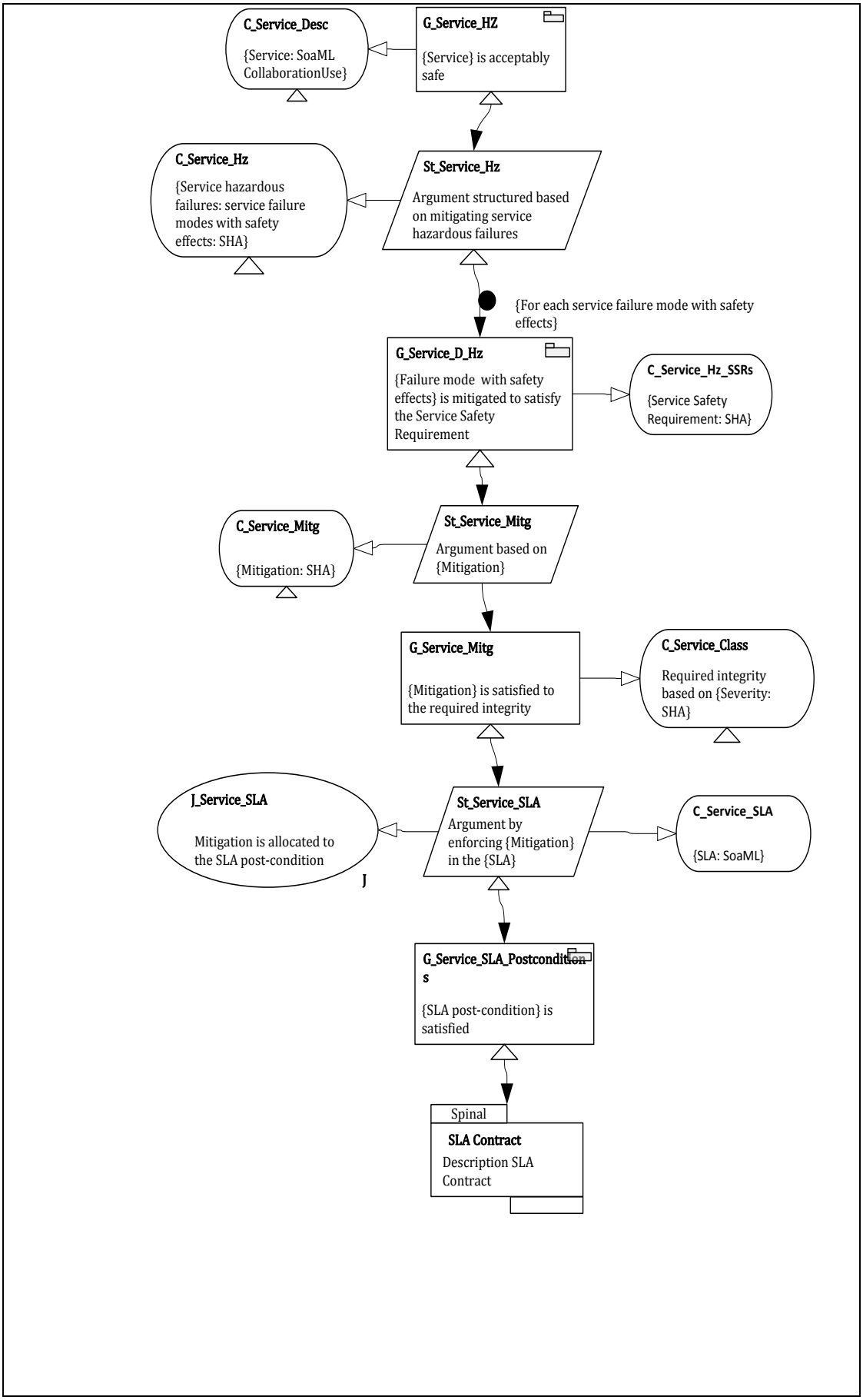
Table 19: Top Argument

5.7.2 Service Argument

This pattern provides justification of the mitigations used to manage the hazardous failures associated with a service, and is structured according to the results of the SHA. The top-level claim is made in the context of the description of the current service and the identified service failure modes that have safety effects. The strategy adopted is to argue the safety of the service by appealing to the adequacy of the mitigations provided for the hazardous service failures, in terms of meeting the service safety requirements. The argument considers each service failure mode that has safety effects in turn. The strategy adopted is to argue that the mitigation is sufficient to achieve the required integrity. The achievement of the mitigation is based on, and supported by, the output or post-condition of an SLA. This is defined in the form of a safety case contract

that mediates the relationship between the Service Argument module and the Participant Argument module.

Service Argument Pattern			
Author	Abdulaziz Al-Humam		
Created	08/07/2014	**Last Modified**	04/07/2015
Intent	The intent of this pattern is to provide an argument to justify the mitigation measures used to manage the hazardous failures for a service and to demonstrate how the reasoning is supported via an SLA Contract Argument.		
Also Known As	N/A.		
Motivation	The contribution of each service to overall service safety needs to be justified. This pattern was created in order to provide this justification, in terms the mitigation of any hazardous contribution from the service.		
Structure			


C_Service_Desc
{Service: SoaML CollaborationUse}
G_Service_HZ
{Service} is acceptably safe
C_Service_Hz
{Service hazardous failures: service failure modes with safety effects: SHA}
St_Service_Hz
Argument structured based on mitigating service hazardous failures
{For each service failure mode with safety effects}
G_Service_D_Hz
{Failure mode with safety effects} is mitigated to satisfy the Service Safety Requirement
C_Service_Hz_SSRs
{Service Safety Requirement: SHA}
C_Service_Mitg
{Mitigation: SHA}
St_Service_Mitg
Argument based on {Mitigation}
G_Service_Mitg
{Mitigation} is satisfied to the required integrity
C_Service_Class
Required integrity based on {Severity: SHA}
J_Service_SLA
Mitigation is allocated to the SLA post-condition
J
St_Service_SLA
Argument by enforcing {Mitigation} in the {SLA}
C_Service_SLA
{SLA: SoaML}
G_Service_SLA_Postconditions
{SLA post-condition} is satisfied
Spinal
SLA Contract
Description SLA Contract

Participants	G_Service_Hz	The top-level claim considers the safety of a specific service. The Service description is provided by the SoaML Service Contract Architecture.
	C_Service_Desc	This context node provides a description of the service within the system. This is provided by the SoaML Service Contract Architecture.
	C_Service_Hz	This context node provides a description of the service failure modes that have safety effects.
	St_Service_Hz	This node presents the strategy adopted to support G_Service_Hz (i.e. mitigation of service hazards)
	C_Service_Hz_SSRs	Definition of the service safety requirements associated with the service. This is provided from the {SSRs: SHA}.
	G_Service_D_Hz	This claim is generated for each failure mode with hazardous effects that has been identified in SHA. This is provided by the {Failure Modes: SHA}.
	St_Service_Mitg	This node presents the strategy adopted to support G_Service_D_Hz, considering the specific mitigation measures used to address a specific hazardous failure. This is provided by the

		{Mitigation_SHA: SHA}.
	C_Service_Mitg	This context defines the list of mitigations for a specific hazardous failure. This is provided by the SHA results.
	G_Service_Mitg	This claim considers the need to achieve mitigation to the level of rigour required, depending on the severity of the failure event. This is provided by the SHA results.
	C_Service_Class	This context node refers to the interpretation of the standard's requirements for integrity level which has been produced for this service. . This is provided in the SHA results.
	St_Service_SLA	The achievement of the mitigation measures is provided by the Participants. This is captured via an SLA in which the post-conditions define the achievement of the mitigation measures.
	J_Service_SLA	The use of SLA post-conditions must be satisfied to enable the mitigation to be achieved and assured.
	C_Service_SLA	This context node provides a reference to the SLA used. This is provided in the SoaML Service Contract Architecture {SLA}.
	G_Service_SLA_Postconditions	This claim argues that the SLA post-conditions corresponding to the

		mitigation are satisfied. Evidence for this is provided by the SoaML Service Contract Architecture {SLA post-conditions}.
	SLA_Contract	The contract can be used to record the interdependencies that exist between the arguments containing the claims about post-conditions and pre-conditions of the SLA. This is provided by the SoaML Service Contract Architecture {SLA post /pre-conditions}.
Collaborations	• C_Service_Hz provides the necessary context for the identification of the safety hazards associated with the service. • G_Service_D_Hz makes a mitigation claim based on the hazardous events described in C_Service_Hz from different perspectives (Service not provided when required; Service provided when not required and incorrect service). • C_Service_Hz_SSRs and C_Service_Class provide necessary context for identifying suitable mitigations of the hazards associated with the service based on the severity level. • G_Service_SLA_Postconditions defines the SLA post condition corresponding to the mitigation within the contract.	
Applicability	This pattern is applicable wherever a service is associated with failures than have safety effects (i.e. hazardous failures).	
Consequences	After instantiating this pattern, the contract module should be developed. The SLA Contract Argument Module for SOA Safety Argument pattern can be used to structure this contract.	
Implementation	Hazard Argument Module should be fully developed and instantiated. Possible problems include attempting to apply the pattern to a system which is not based on SOA, and	

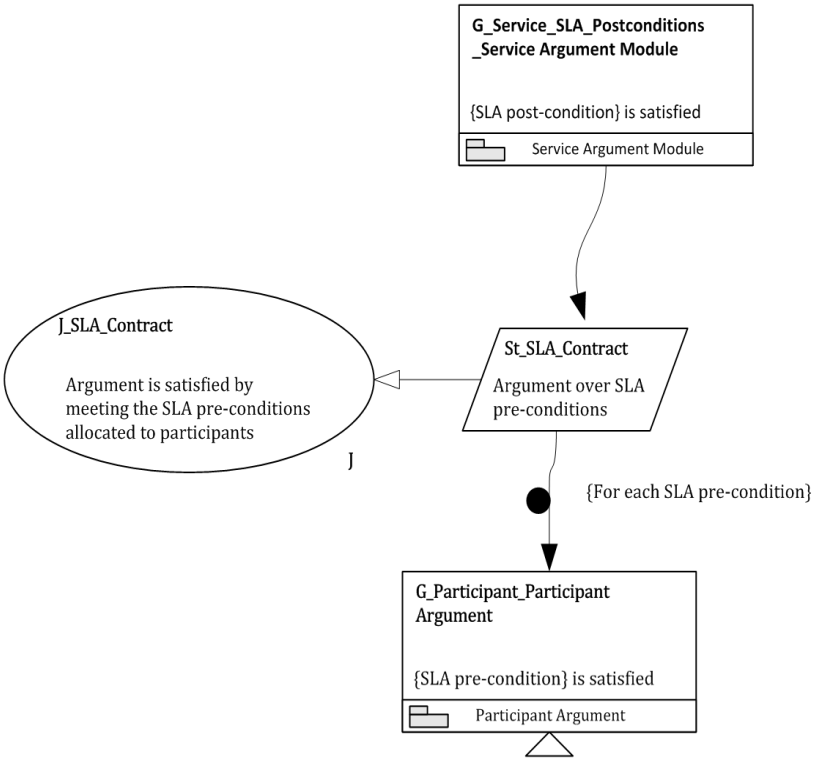
	attempting to apply the pattern without having defined safety contract interfaces.
Examples	See the two case studies in this thesis.
Known Uses	See examples above.
Related Patterns	SLA Contract Argument Pattern.

Table 20: Service argument Pattern

5.7.3 SLA Contract Argument

It Chapter 4, we explained how safety analysis can help identify the safety requirements for each service. The SLAs for each service can specify the pre- and post-conditions that are necessary to ensure that these requirements are met. The SLAs define the basis for argument contract pattern in our SOA safety argumentation approach. Table 21 shows the documentation of the pattern. The top goal of the pattern is an away goal which claims that the SLA post-conditions from Service Argument Pattern are satisfied. The strategy adopted is to argue over SLA pre-conditions. The justification for this is based on meeting the SLA pre-conditions allocated to participants. Away goals are used to argue that one or more participants satisfy the SLA pre-conditions.

SLA Contract Argument Pattern			
Author	Abdulaziz Al-Humam		
Created	08/07/2014	**Last Modified**	04/07/2015
Intent	The intent of this argument contract pattern is to provide the justification that is needed to link the claims about the services and those about the participants consuming and providing these services. This pattern can be used to record and justify the interdependencies that exist between post-conditions and pre-conditions for the services and their associated participants.		
Also Known As	SOA Safety Case Contract		
Motivation	Just as SLAs are used to record interdependencies between services within SOA design, safety case contract modules		

	are used to represent rely-guarantee relationships between modules in modular safety case applications. This pattern was developed to exploit the correspondences between the structure of the SOA design and the organisation of the safety case and therefore improve clarify and maintainability.	
Structure 		
Participants	G_Service_SLA_postconditions_ Service Argument Module	The top-level claim states that the SLA post condition is satisfied. The claim referenced in the away goal here originates in the Service Argument Module. It is used to show sufficient support to satisfy the requirements for mitigation. The information to instantiate this goal is provided by

		the SoaML Service Contract Architecture {SLA post-conditions}.
	St_SLA_Contract	This node presents the strategy of decomposition of the argument over the SLA pre-conditions that must be satisfied by the participants.
	J_SLA_Contract	This node justifies the decomposition strategy by claiming that the use of SLA pre-conditions – satisfied by the participants – is sufficient to enable the achievement of the SLA post-conditions.
	G_Participant_Participant Argument	This claim states that the pre-condition is satisfied by a participant. This is made in the form of an away goal that references a goal supported by argument and evidence within a Participant Argument. This is provided by the SoaML SLA pre-conditions.
Collaborations	• G_Service_SLA_postconditions_Service Argument Module and G_Participant_participant Argument are away goals that provide the post/pre conditions necessary to satisfy the safety requirements (contract) within the SLA service of an SOA system. • J_SLA_Contract provides the justification for the satisfaction of the pre-conditions allocated to participants as a means for meeting the SLA post-	

	conditions.
Applicability	This pattern is applicable wherever safety arguments need to be established for the use of SLA within an SOA safety application. It assumes that both post- and pre- conditions are explicitly defined for the SLA.
Consequences	After instantiating this pattern, the goals referenced in the away goals will be developed within the Participant Argument Module (G_Participant_H).
Implementation	Possible problems include attempting to apply the SLA contract pattern to a system which is not based on SOA, attempting to apply the SLA contract pattern without having defined safety contracts and attempting to apply the SLA contract pattern without sufficient SLA information from the SoaML/BPMN models.
Examples	See the two case studies in this thesis.
Known Uses	See examples above.
Related Patterns	Participant Argument Pattern.

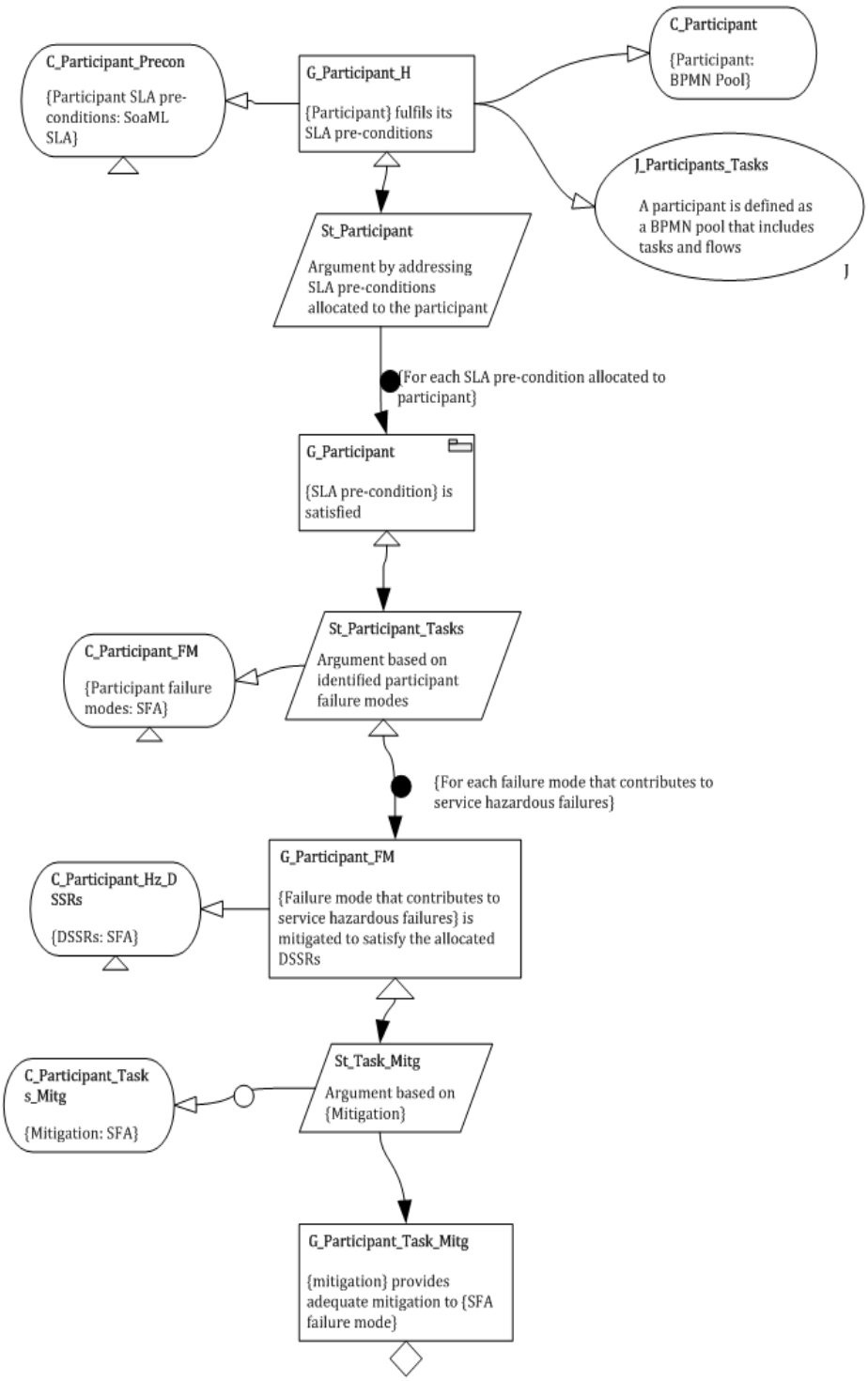
Table 21: SLA Contract Argument Pattern

5.7.4 Participant Argument

This pattern provides the justification basis for the behaviour of a Participant in meeting SLA pre-conditions allocated to it. Table 22 shows the documentation of the Participant Argument pattern in GSN. The instantiation of this pattern starts with the goal G_Participant_H, which is a claim that the participant meets its allocated pre-conditions. The structure of the pattern is driven by the results of the SFA. The claim in G_Participant_H is made in the context of the Participant definition (specified in BPMN) and the allocated SLA pre-conditions (in the SoaML model). The strategy adopted is to argue over each SLA pre-condition allocated to the participant. The sub-claims demonstrate how the allocated SLA pre-conditions referenced in the SLA Contract Argument Pattern (G_Participant _Participant away goal) are satisfied. The strategy adopted here is to argue over the identified Participant failure modes which are associated with tasks and their flow interactions. This is made in the context of the description of the Tasks and their interactions form BPMN

model and the Participant failure modes (from the SFA results). A Participant is defined in the form of a BPMN pool that includes tasks and flows. The rest of the argument is used to show how the participant failure modes are mitigated by appealing to the results of the SFA.

Participant Argument Pattern			
Author	Abdulaziz Al-Humam		
Created	08/07/2014	**Last Modified**	04/07/2015
Intent	The intent of this pattern is to provide justification for the participant behaviour and to create an argument regarding the safety of the flow between the tasks implemented by a participant in an SOA system. The safety of the participant's behaviour is argued from the perspectives of both the SLA pre-conditions and the participant interaction flow.		
Also Known As	Interaction behaviour pattern		
Motivation	The motivation for this pattern is the need to justify the behaviour of a participant, as implemented in the form of tasks and flows, and to demonstrate how it satisfies its allocated SLA pre-conditions.		
Structure			



Participants	G_Participant_H	The objective of the argument is to provide sufficient support for the claim that the {Participant} fulfils its {SLA pre-conditions}. The information is provided by the BPMN model (pool) and the

		SoaML model (SLA pre-conditions).
	C_Participant	This context node defines the by the BPMN model: pool.
	C_Participant_Precon	This context node provides a description of the SLA pre-conditions allocated to the Participant. This is provided by the SoaML model.
	J_Participants_Tasks	This explains that a Participant is defined in the form of a BPMN pool.
	St_Participant	The argument strategy is based on addressing all of the pre-conditions allocated to the Participant.
	G_Participant	This claim is instantiated for each pre-condition, and states that a pre-condition allocated to the participant as a safety requirement is satisfied. This is provided by the SoaML: SLA pre-conditions.
	St_Participant_Tasks	This node describes the mitigation strategies for the identified failure modes of the {Participant}. This is provided by the BPMN model.
	C_Participant_FM	This context node provides a description of the failure modes associated with the Participant. This is provided by SFA: failure modes.
	G_Participant_FM	This node provides a claim that each failure mode is sufficiently mitigated. This is

		provided by SFA: deviation.
	C_Participant_Hz_DSSRs	This node refers to the safety requirements associated with the Failure mode that contributes to service hazardous failures. This is provided by {DSSRs: SFA}.
	St_Task_Mitg	This node describes an argument strategy by which the adequacy of each mitigation is demonstrated. The mitigations are implemented from SFA: mitigation.
	G_Participant_Task_Mitg	This claim considers the adequacy of the selected mitigation from SFA: Failure modes.
	C_Participant_Tasks_Mitg	This context node defines the list of tactics to provide suitable mitigation for a specific failure mode: e.g high-integrity, redundancy, diversity and monitoring/checking. This information is provided by SFA: mitigation of SFA.
Collaborations	• C_Participant and C_Participant_Precon provide necessary context for the system's interaction with the Participant, specifically for SLA pre-conditions. • G_Participant_H and G_Participant argue that the pre-conditions relating to {Participant} have been satisfied. • C_Participant_FM provide necessary context for identifying the interaction and the failure modes associated with the Participant. • G_Participant_Task_Mitg considers the suitable mitigations used for addressing Participant failure	

	modes.
Applicability	This pattern is applicable wherever participant safety arguments need to be established for the use of a SOA-based safety critical system.
Consequences	After instantiating this pattern, one undeveloped goal will remain: G_Participant_Task_Mitg. The argument supporting this must be developed, based on more detailed implementation information.
Implementation	Possible problems include attempting to apply the pattern without sufficient design information (implementation) from the SoaML/BPMN models.
Examples	See the two case studies in this thesis.
Known Uses	See examples above.
Related Patterns	None provided at this stage.

Table 22: Participant Argument Pattern

5.8 Healthcare Case study: SOA Safety Case

This section presents the results of applying the SOA safety argument pattern catalogue outlined in the previous section to the healthcare case study introduced in Chapter 4. The purpose of this part of the case study is to examine how the SOA safety argument pattern catalogue can help in communicating the justification for the safety of the ambulance system and its associated services. It also shows how the analysis process applied to the healthcare case study in Chapter 4 relates to the overall structure of the SOA safety argument. The case study will also be used to illustrate how the safety requirements and mitigations arising from the analysis can be used to define the safety argument contracts based on the SLAs defined during the design.

The description of the system and the system design including all design models are taken directly from the Health and Social Care Information Centre (148) which was presented in Chapter 4. This section considers the safety arguments for one specific service: Dispatch Ambulance. Additional safety arguments are described in Appendix D.

5.8.1 Ambulance Service System Safety Argument

The safety case for the Ambulance Service System starts with an instantiation of the Top Argument pattern. The Top Argument includes a service-directed argument, which covers the main safety claims concerning the ambulance service system services.

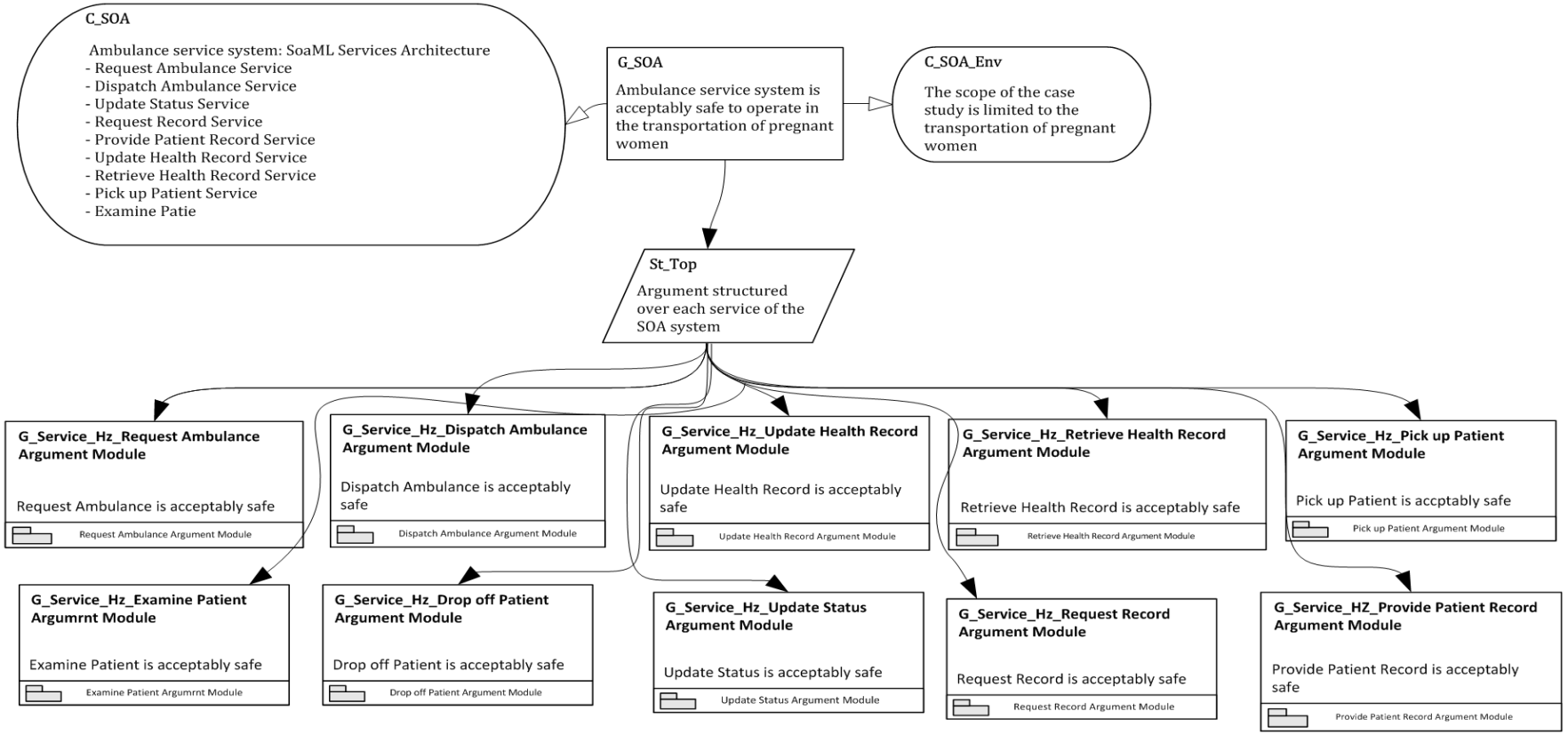


Figure 46: Top Argument Module for Dispatch Ambulance

The results of the instantiation are shown Figure 46. G_SOA presents the claim about the overall safety of the Ambulance Service System in the context of the use case, i.e. the transportation of pregnant women (*C_SOA_Env*) and the list of services covered (C_SOA). The strategy decomposes the argument over all the identified services within the system, based on the SOA specification defined in SoaML. For each service, the argument makes a claim that the service is acceptably safe. This is done by means of creating an away goal for each service which references the corresponding Service Argument module. In this section, we only focus on the argument for the Dispatch Ambulance service.

5.8.2 Service Argument for the Dispatch Ambulance Service

The argument structure in Figure 47 shows the instantiation of the Service Argument pattern for the Dispatch Ambulance service. The instantiation is based on the result of the SHA analysis of the Dispatch Ambulance service we presented in Chapter 4. The top-level claim for the Dispatch Ambulance states that the service is acceptably safe, in the context of the SOA specification defined in SoaML CollaborationUse. The strategy argues over all the identified service hazards within the service based on the analysis carried out in the SHA. This strategy is presented in the context of the results of the modelling and SHA for this service.

The argument focuses on three failure modes which have hazardous effects, which have been generated from the SHA results, namely:

- Ambulance not dispatched;
- Ambulance dispatched later than intended;
- Ambulance dispatched to the wrong address.

The argumentation strategy adopted for each failure mode is to argue that the failure mode has been mitigated to the required level of integrity. For example, the failure 'Ambulance not dispatched' is mitigated by active monitoring and cross-checking between requested and dispatched ambulances. The mitigation claim is then defined in the form of a safety contract post-condition that is supported by the SLA Contract Argument.

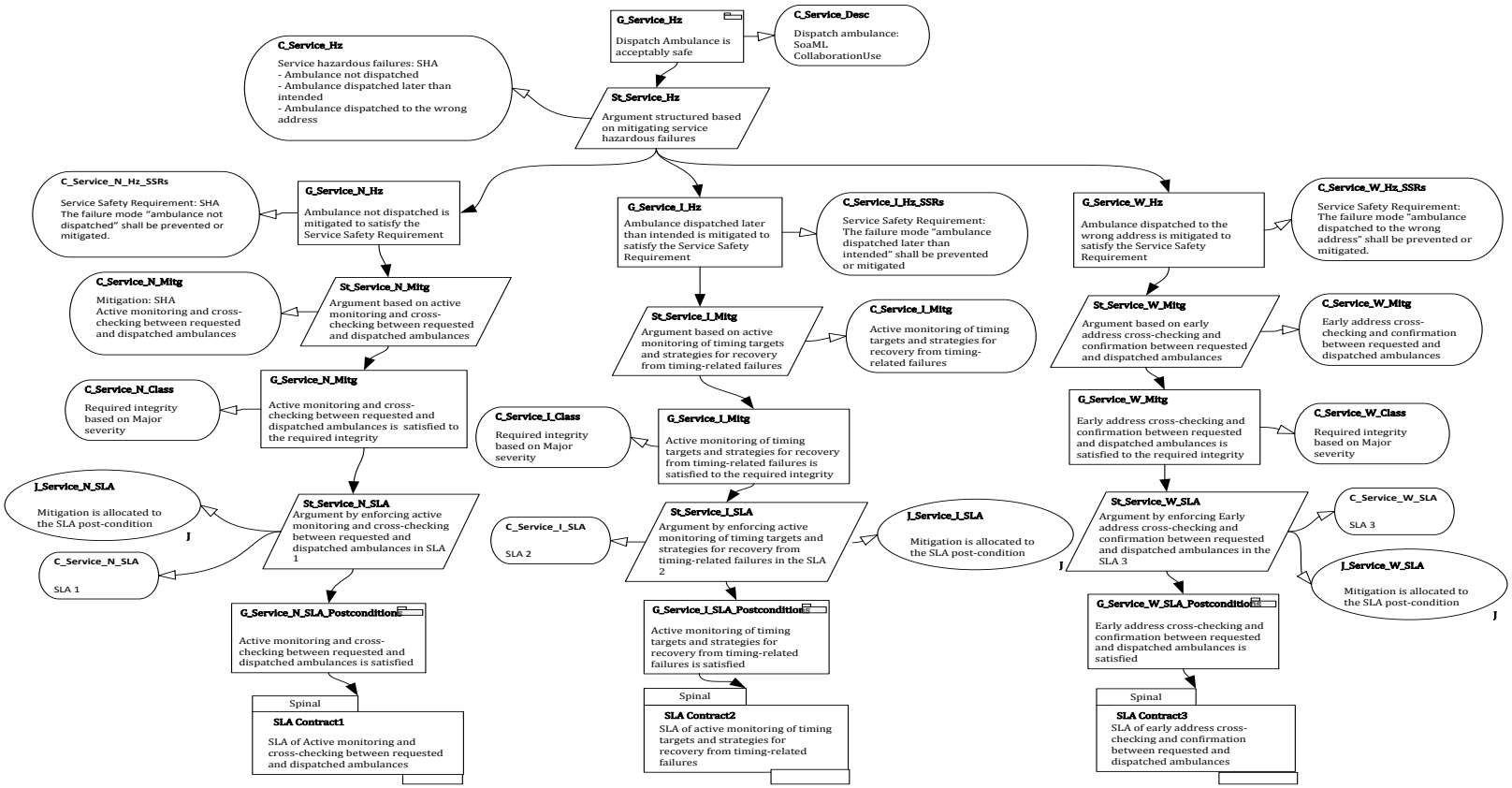


Figure 47: Service Argument for Dispatch Ambulance

5.8.3 SLA Contact Argument for the Dispatch Ambulance Service

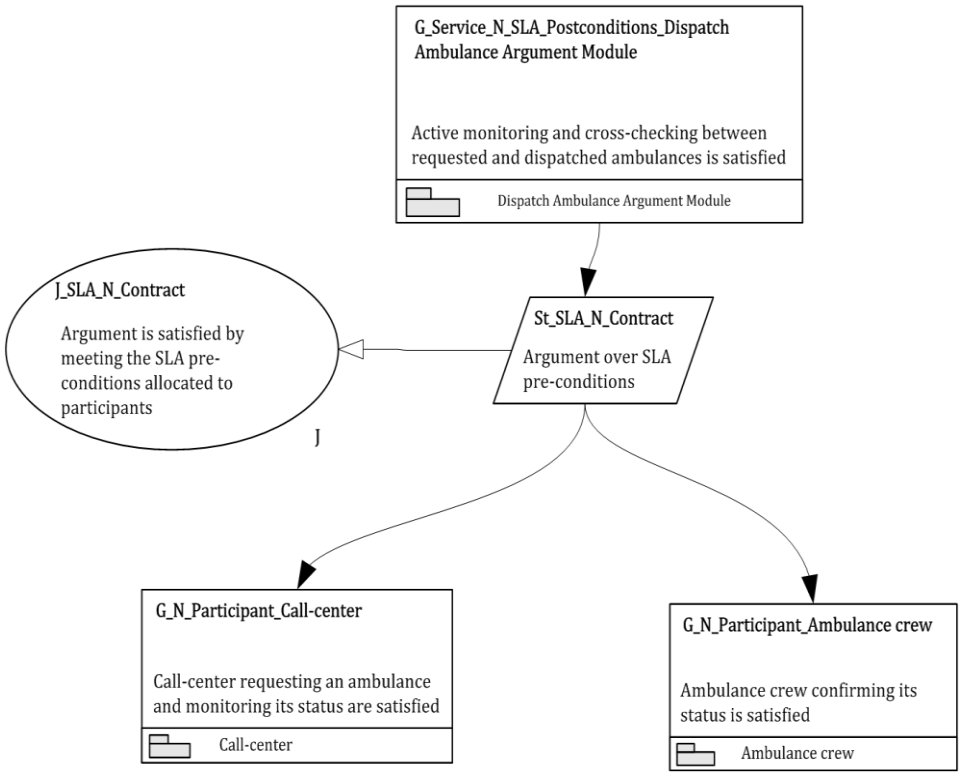


Figure 48: SLA contact Argument for Dispatch Ambulance

Figure 48 shows the instantiation of the SLA Contract Argument pattern for the Dispatch Ambulance service. The top-level away goal states that active monitoring and cross-checking between requested and dispatched ambulances are satisfied. These correspond to the post-conditions of the SLA associated with the Dispatch Ambulance service. The argument strategy is to argue over the satisfaction of the set of SLA pre-conditions. This takes the form of away goals that reference claims stated in the Participant Argument for the participants satisfying the SLA pre-conditions. For example, the away goal which points to the Call-Center argument module references a claim in that module which states that the Call-centre sends a request to the ambulance crew and then monitors its status.

5.8.4 Participant Argument for the Call-Center Participant

Figure 49 shows the instantiation of the Participant Argument pattern for the Call-Center Participant. The argument is based on the satisfaction of the pre-conditions allocated to the Call-Center Participant via the SLA in which the Participant is involved; namely, that the Call-Center sends a request to ambulance crew and monitors its status. The argument then provides a justification for how the above claims are supported based on the results of the SFA and modelling in BPMN.

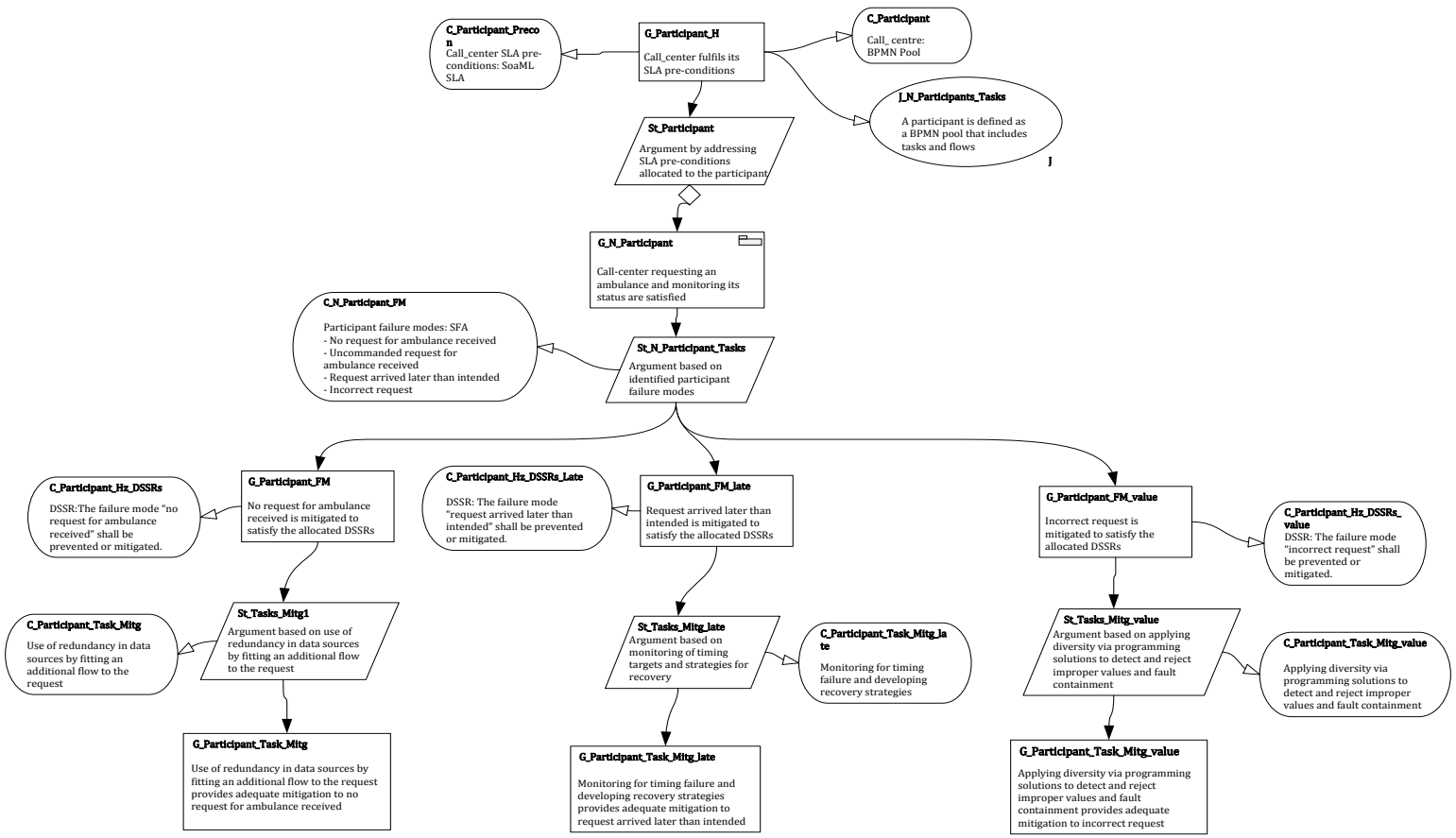


Figure 49: Participant Argument for Dispatch Ambulance

5.9 Summary:

In this chapter, we have presented a catalogue of safety argument patterns for justifying the use of SOA in safety-critical applications. The overall safety argument is created in a modular manner to ensure a clear correspondence between the structure of the safety argument and the organisation of the SOA design. These argument patterns link the safety argument for the SOA with the results of the modelling in SoaML and BPMN and the results of the safety analysis on SHA and SFA. Finally, we illustrate the use of the safety argument patterns for one of the services within the healthcare case study.

Chapter 6

6. Evaluation

6.1. Introduction

This chapter presents the results of the evaluation of the SOA Safety Assurance Framework developed in this research. It revisits the research hypothesis and explains the means for evaluation used in this thesis. Further, the chapter provides the details of the second case study we performed and the results of the evaluation with respect to the research contributions.

6.2. Research Hypothesis

The hypothesis proposed in this thesis is as follows:

Integrated *architectural modelling and analysis of safety-critical service-oriented systems provides a* ***traceable****,* ***consistent*** *and* ***systematic*** *means for* ***assuring*** *the safety of these systems.*

Below is an explanation of the main terms used in the hypothesis.

- **Integrated:** safety analysis and design decisions evolve together;
- **Traceable:** forward and backward model-based traceability between design and safety analysis is provided;
- **Consistent:** design features are reflected in the safety analysis and the safety assurance and vice versa;
- **Systematic:** a methodical approach to performing safety assurance is proposed;

- **Assuring:** the approach provides an explicit means for reasoning about the safety properties and system behaviour.

The above terms define the basis for the criteria used for evaluation of the SOA Safety Assurance Framework.

This hypothesis is supported by the following four contributions made as part of this PhD research:

- Identification and configuration of SOA modelling notations, namely SoaML and BPMN, based on two criteria: coverage of the structure and behaviour of the services which is deemed suitable for the safety analysis of SOA.
- Development of safety analysis techniques for SOA, namely SHA and SFA, which can be used to examine the hazardous failures of services and the failure behaviour of lower-level implementations.
- Development of the concept of modular safety cases for the assurance of safety-critical SOAs, promoting correspondence between the structure of the SOA and the organisation of the modular safety case. This generic argument approach is defined in the form of an SOA Safety Argument Pattern Catalogue, which can be instantiated to produce safety arguments for specific SOA-based systems.
- Implementation of tool-support that provides automated capabilities for building the SOA models, applying the safety analysis and building the safety case for the system.

6.3. Means of Evaluation

Two case studies have been carried out during this research. The first case study, which was presented in Chapters 4 and 5, is based on the healthcare domain. The second, which is presented later in this chapter, is based on the oil and gas domain. The research was also evaluated through a peer-reviewed publication and semi-structured interviews with practitioners.

6.3.1. Case Studies

The two case studies provided the primary means for evaluating the technical criteria related to the research hypothesis, namely that the proposed approach

should be ‘integrated’, ‘traceable’, ‘consistent’, ‘systematic’ and ‘assuring’. These are qualitative criteria and the results of examining these criteria are discussed in Section 6.6. The second case study in particular provides the basis for evaluating the feasibility of the SOA Safety Assurance Framework developed in this thesis.

The two domains of the case studies, healthcare and oil and gas, are typically categorised as safety-critical and tend to be structured around services. Nevertheless, they have important differences. Safety in Oil and Gas is well established and is heavily regulated. The environment also tends to be relatively well understood. Safety in healthcare on the other hand presents different kinds of challenges. The environment is harder to control and safety as a discipline, especially for IT systems, is not well defined (149). The concept of engineering risk in healthcare is not typically subjected to systematic safety analysis, as compared to Oil and Gas. Each case study provides a different treatment of risk associated with services in the corresponding domain. In healthcare for example, safety analysis tends to be high level and not as detailed as in a traditional safety-intensive domain such as Oil and Gas.

6.3.2. Semi-Structured Interviews

In order to provide an independent evaluation of our approach, we conducted semi-structured interviews with domain experts from both the healthcare and oil and gas domains. The results of this evaluation are discussed in section 6.5 below.

6.3.3. Peer Review

The technical accuracy of the research was also evaluated through a peer-reviewed publication published in the *Medical Cyber Physical Systems Workshop, 2014*.

The work was also evaluated by means of presentations and interactions with members of the High-Integrity Systems Engineering (HISE) research group at York.

6.4. Industrial Case Study: Natural Gas Processing

This section introduces our second case study, in which the SOA Safety Assurance Framework was used to model an industrial service-based system. The case study was intended to demonstrate the application and feasibility of the safety analysis methods and safety case approach defined in this thesis.

6.4.1. Aim

The main aim of this industrial case study, based on an example from the oil and gas domain, is to examine the feasibility of applying our SOA Safety Assurance Framework in the light of the technical criteria identified in the research hypothesis, namely: 'integration', 'traceability', 'consistency', 'systematicness' and 'assurance'. The results of the application of the SOA Safety Assurance Framework will be compared to the traditional safety analysis approach based on HAZOP, the primary safety analysis technique used in the oil and gas industry.

6.4.2. System Description

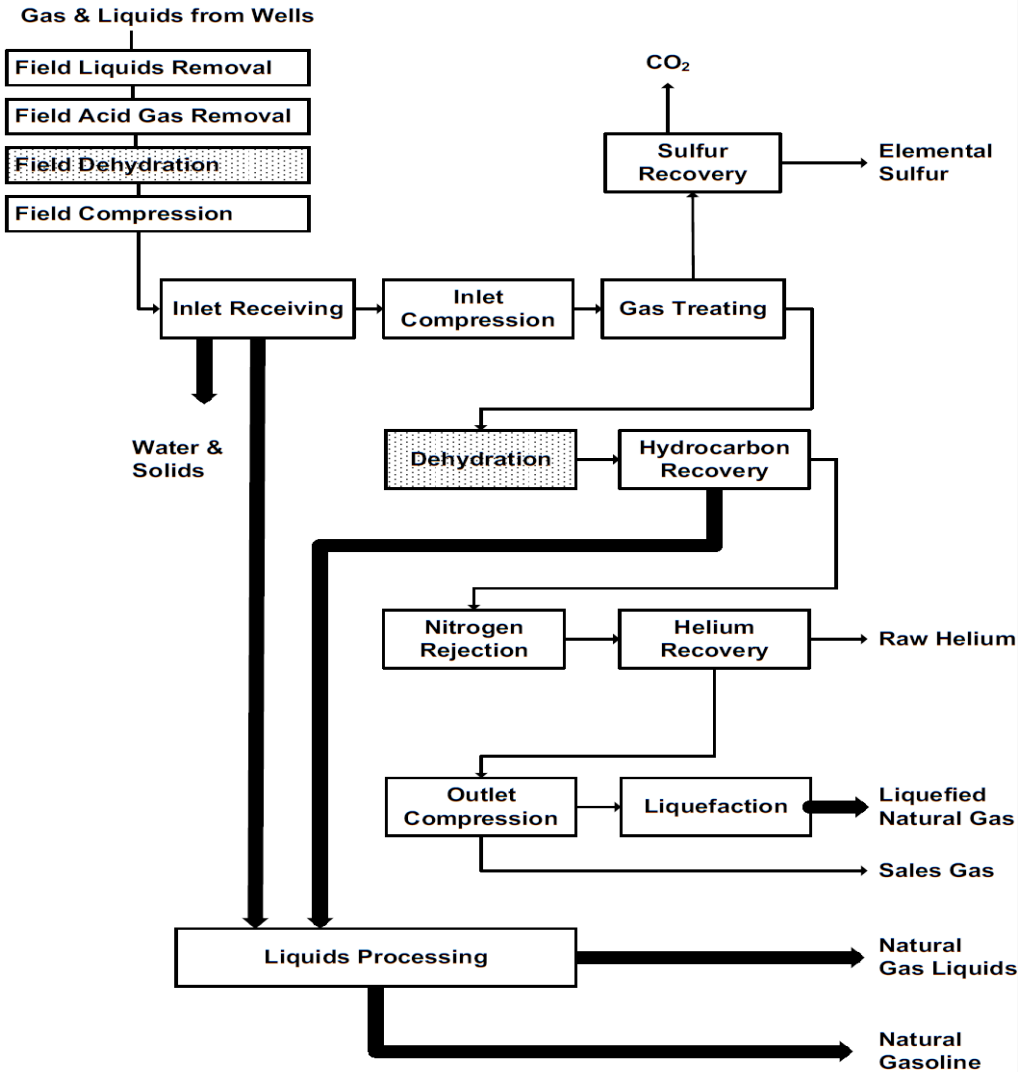


Figure 50 Overview of Gas Plant Processes (150)

The case study is based on part of a system that is used to produce liquid gas in a Natural Gas Processing Plant. Natural Gas Processing is a commercially sensitive activity, and although this case study was developed in collaboration with a chemical engineer from the oil and gas domain, the system and plant used in this case study do not refer to any actual industrial systems. Some of the system descriptions and analyses used in this case study are based on information in the published literature, mainly (150) and (151).

Figure 50 describes the Natural Gas Processing system, which comprises seven main services. These have been used to provide the detailed organisation of the SOA system:

(1) Supply Natural Gases;

(2) Separate Gases;

(3) Absorb Water;

(4) Heat Exchange;

(5) Separate Gas from Liquid;

(6) Send Emergency Signal;

(7) Call Emergency Number.

The Gas Processing Plant contains many processing units that provide facilities to receive the natural gas, filter it, separate it, cool it, and store the gas and liquids in a tank for shipment. The functions of the Plant comprise primary processes such as gas treatment, dehydration, acid gas removal, as well as heating and cooling media such as steam, cooling water and chilled water.

The gas is provided from the wells to a treatment unit. The treatment unit separates the gas into acid and sweet gas. The acid gas contains hydrogen sulphide (H_2S) and carbon dioxide (CO_2). The sweet gas contains C_1, C_2, C_3, C_4, C_5 and H_2O in gas phases. The gas stream enters the dehydration unit at a temperature of 120°F where it first goes to dehydration unit. This unit contains a chemical catalyst that helps in removing the water from the gas. The objective is to avoid freezing the gas stream inside the pipe. The performance of the dehydration unit is monitored by a pressure transmitter. A transmitter is a device that converts measurements from a sensor to a signal which is received when the high pressure inside the unit results in changes to the catalysis process. After the water is removed, the gas goes to a cooling unit where it is cooled from 120°F to –40°F. The change in temperature is monitored by temperature transmitters. A transmitter (pressure or temperature) should send a signal to the control operator inside the control room, allowing him to monitor the pressure or temperature changes. For any change above or below a pre-defined limit, the control operator should call a supervising operator to decide on any remedial action needed. On leaving the cooling unit, the gas is passed to a separator unit to separate the gas from the liquid. The drop in temperature helps to condense the gas into liquid form. The liquid stream is called liquid gas. After processing, the liquid gas is shipped to an

external chemical manufacturing company, whereas the gas is supplied to an electricity company.

The performance of the unit as a whole is monitored by a human control operator, who monitors indications of pressure and temperature levels in the gas and liquid gas which are sent through transmitter instrument devices from the processing area to the operator's computer. The control operator is responsible for detecting process changes and for advising the outside operator accordingly through a radio communication channel.

In the event of a gas leak, the control operator is expected to detect a sharp increase in the temperature based on the cooler's indicating that no gas is passing through. The upstream emergency isolation valve should close automatically, since it has a logic where any sharp increase in temperature should lead to the valve closing. This is to avoid the continuation of gas leakage from the source. The leaking gas is a fuel and any spark of ignition is likely to result in a fire. Next, the control operator should call the emergency centre, notifying the fire, medical and security services. The fire services are on continuous standby, and should be ready to extinguish any potential fire resulting from the leak. Similarly, the ambulance team should be ready to provide the necessary medical support, since there is a potential for people to inhale toxic gas which could result in death or health complications. The security team are responsible for traffic control and for ensuring that authorized people can access the emergency location (where the leak took place) (150).

6.4.3. Overview of Oil and Gas Safety

Before applying our SOA Safety Assurance Framework to the natural gas processing system, we will use this section to introduce the key safety factors of the oil and gas domain.

An early activity in any typical safety analysis is the identification of hazards. The COMAH Safety Report Assessment Manual (SRAM) (152) provides some indication of the types of hazards relevant to the oil and gas industry and methods used to identify them. The principal methods they identify are:

- Hazard and operability studies (HAZOP);

- Safety review and studies of the causes of past major accidents and incidents'
- Industry standards or bespoke checklists for hazard identification;
- Failure mode and effect analysis (FMEA);
- Job safety analysis (e.g. task analysis);
- Human error identification methods.

According to the SRAM, the analysis should first develop a clear understanding of the domain (site operations), the materials involved and the process conditions. Secondly, it must identify the hazards to people on-site and off-site and to the environment. Finally, there must be an analysis of different ways in which the hazards can be eliminated, reduced in scale, realised and controlled (153).

6.4.4. Accidents and Hazards Identification

It is a high-level safety requirement for the process that the hazard identification exercise should identify all foreseeable major accidents. In order to make the risk assessment feasible, it is possible to group potential accident scenarios together for analysis. If this is the case, the grouping should be conservative in terms of the risk associated with each group, and should include worst-case scenarios. We apply this principle in our case study by identifying and classifying the accidents, hazards and failure modes. The main types of accident in this domain are: Explosion, Serious Damage, Major Leak and Fire.

6.4.4.1 Accident Identification in the Oil and Gas Domain

Explosion can be defined as a quick increase in volume and release of energy in an extreme manner, generally with the generation of high temperatures or overpressure and the release of gases or liquids (154). Large explosions are capable of causing immediate critical damage, e.g. BP Deepwater Horizon Oil Spill and offshore drilling in the Gulf of Mexico (155).

Accidents of the serious damage type can happen when corrosive chemicals are used in equipment which is not designed to tolerate them. Corrosion is a condition which means that an agent that can destroy and damage other substances when it comes into contact with them, e.g. mixing incompatible

chemicals because of pressure reversal (incompatible materials may be mixed with each other, leading to corrosion of the vessel containing them).

Major fire is defined as a quick oxidation of a material in the exothermic chemical process of combustion, releasing heat, light, and various reactive products (154). One recent example of a major fire accident is the fire at the Gregory natural gas plant in Texas in 2013, which resulted in a large blaze and operational loss (156).

In the gas domain, 'Major Leak' is defined as an escape of natural gas from a pipeline or other containments to a living area or any other where the gas must not be present (154). For example, the Bhopal disaster was a leak of gas and other chemicals which took place at a plant in India in 1984, resulting in the exposure of up to 15,000 people to an extremely harmful chemical.

6.4.4.2 Hazard Identification

Explosions may occur in many different ways. The contributory hazards can typically be classified in terms of factors such as High pressure and Blockage. Failure modes can occur in many different environments and configurations (157):

- chemical reaction inside the vessel, blocked outlets (blocked-in pump and blocked-in compressor);
- loss of cooling capacity (failure of air-cooled heat exchanger, failure of water-cooled heat exchanger);
- abnormal heat input (excessive heat added);
- electrical power failure (electrical power may lead to high pressure e.g. loss of power will cause the vapours to pass thought the condenser without being cooled or condensed, thus creating a high-pressure situation and control failure or valve failure).

One noteworthy cause of high pressure is close instantaneous evaporation of liquid through contact with hotter liquid, which can result in an explosion. This may occur if liquid gas comes into contact with water in areas where there is a potential ignition source (a key focus in our case study). The very high liquid gas pressure can lead to an explosion. Such an event might occur during liquid gas transfer between an onshore terminal and the liquid gas carrier (157).

Serious damage may be caused by hazards such as incorrect valve closure or a failure to close valves when required (mainly human error). Another potential example is the supply line containing water that is either not flowing or is flowing very slowly. In this case, an ambient temperature of below freezing point can cause the water in the line to freeze. This may create serious damage within the line where the flow is stopped.

Major fire can also be caused in many different ways. The hazards associated with fire can be classified based on the flammability of the gas/liquid mixtures. If air enters a process involving flammable hydrocarbons and/or reducing chemicals then a serious accident could happen, or could occur afterwards as a in the presence of an ignition source. For example, if air enters the process through open lines and vessels, this could result in unwanted oxidation processes and air ingress, leading to formation of an explosive mixture (157).

6.4.5. SOA Modelling

In this section, we apply SoaML and BPMN to modelling the structure and behaviour of the Gas Processing system. The diagram in Figure 51 provides a structural description of the system in terms of its services and participants. This is defined using the SoaML Services Architecture diagram. Specifically, it focuses on the Natural Gas Processing System architecture with eight participants that are connected together by seven services.

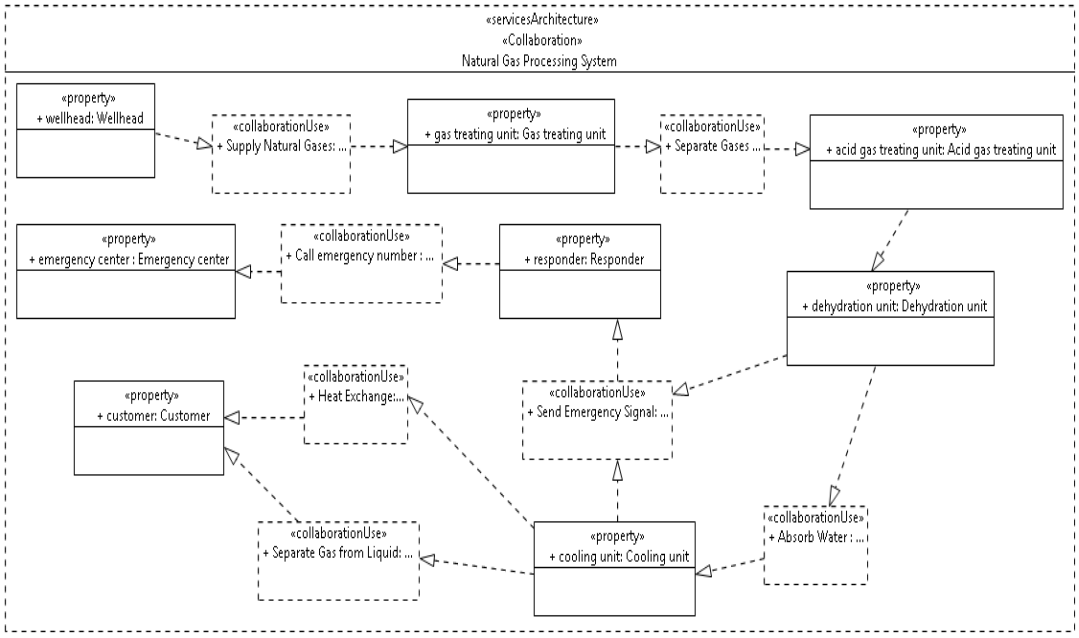


Figure 51: SoaML Service Architectures Model for Gas Processing

Figures 3 to 6 show the different types of interface that the SoaML standard (158) provides for gas modelling (Consumers/Providers). It shows the interfaces associated with the services. The consumer and provider participant roles are modelled based on the defined interfaces. This provides detailed traceability between the SoaML models and the behavioural model in BPMN where each SoaML interface operation is mapped into a BPMN task.

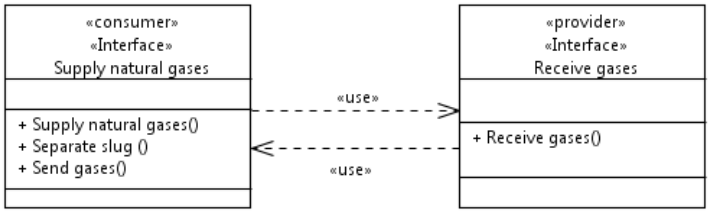


Figure 52: Interfaces with Operations for the Supply Natural Gases Service

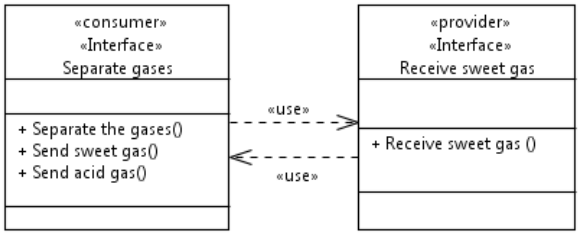


Figure 53: Interfaces with Operations for the Separate Gases Service

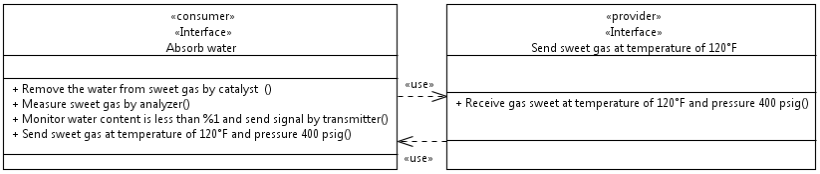


Figure 54: Interfaces with Operations for the Absorb Water Service

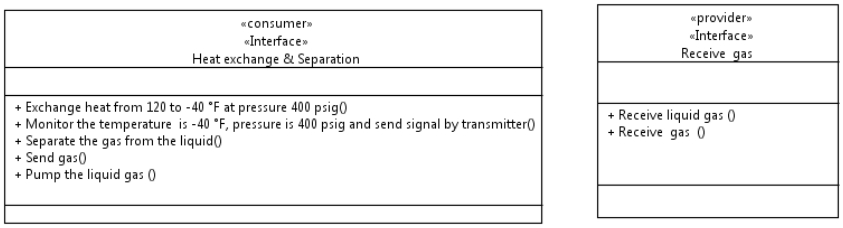


Figure 55: Interfaces with Operations for Heat Exchange and Separate Gas from Liquid Services

Figure 56 presents a more detailed behavioural model of the SOA design using BPMN, with the core activities and execution flow shown as processes. For

example, eight BPMN pools are created to capture the tasks allocated to the participants in the SoaML model in Figure 51. Furthermore, the BPMN model describes in more detail the treatment of sweet gas through providing liquid gas and steam gas.

In the next section, we describe how the SoaML and BPMN models provide the basis for performing the SOA safety analysis using SHA and SFA.

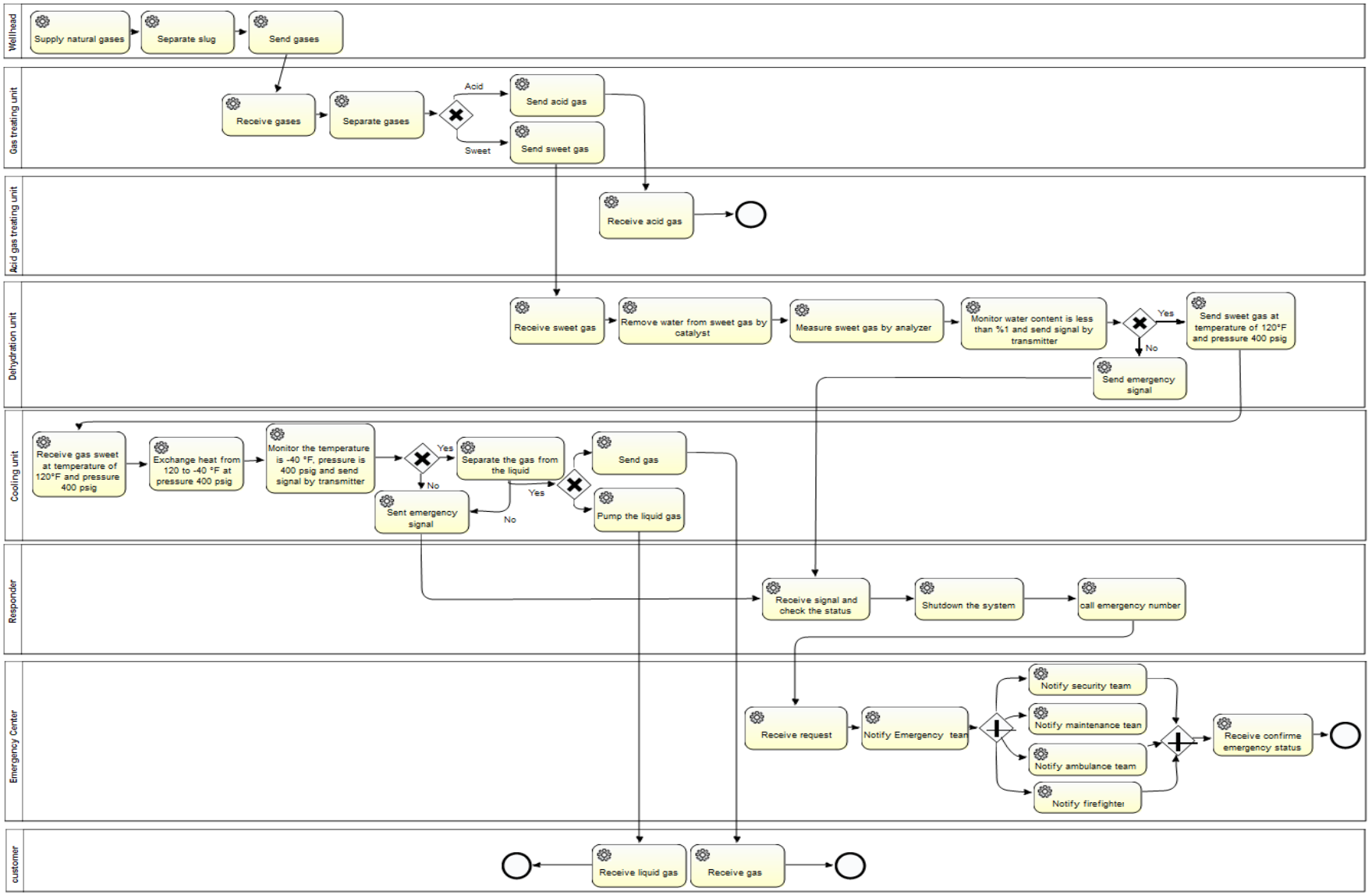


Figure 56: Gas Processing BPMN Model

6.4.6. SOA Safety Analysis

In this section, we show how we applied the SHA and SFA methods to the Natural Gas Processing case study that we introduced in Section 6.4.2. The objective of this part of the case study is to illustrate the use of the methods and provide an evaluation of the feasibility of the analysis approach.

6.4.6.1. SHA

In this section, we provide an overview of the four main services used in the case study with their SLAs and describe how SHA is applied to the services within the system. The services used in the case study are as follows:

- Absorb Water;
- Heat Exchange;
- Separate Gas from Liquid;
- Send Emergency Signal.

The following subsections provide an overview of these services and of the analysis results.

Absorb Water Service

In its gaseous form, natural gas contains H_2O. When converting to a liquid, the H_2O levels in natural gas should be decreased to less than 1% in a physical absorption process in which the gas is contacted through a liquid that preferentially absorbs the water vapour. The liquid solvent must have the following properties in order for the absorption process to be completed (150):

- a high affinity for water and a low affinity for hydrocarbons;
- a low volatility at the absorption temperature to reduce vaporisation losses;
- a low viscosity for ease of pumping and good contact between the gas and liquid phases;
- a good thermal stability to prevent decomposition during regeneration; and
- a low potential for corrosion.

In practice, glycols such as ethylene glycol (EG), diethylene glycol (DEG), triethylene glycol (TEG), tetraethylene glycol (TREG) and propylene glycol are the most commonly used absorbents. Triethylene glycol is the glycol of choice in most instances.

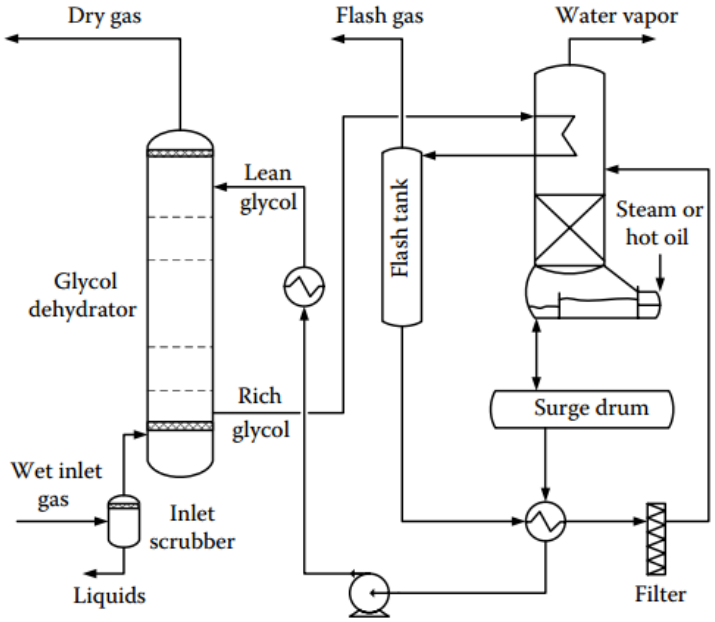


Figure 57: Schematic of a Typical Glycol Dehydrator Unit (159)

Figure 57 shows an example of the use of glycol for H2O absorption. The wet gas passes through an inlet scrubber to remove solids and free liquids, and then enters the bottom of the glycol dehydrator. Gas flows upward in the dehydrator, while lean glycol solution (glycol with little or no water) flows down over the trays. Rich glycol absorbs water and leaves at the bottom of the column while dry gas exits at the top. The rich glycol flows through a heat exchanger at the top of the still where it is heated and provides the coolant for the still condenser. Then the warm solution goes to a flash tank, where the dissolved gas is removed. The rich glycol from the flash tank is further heated by heat exchange with the still bottoms, and then becomes the feed to the still. The still produces water at the top and a lean glycol at the bottom, which goes to a surge tank before being returned to the contactor (150).

The catalyst works like a sponge (known for their exceptional ability to absorb water). The gas stream enters the dehydration unit at a temperature of 120 °F and first goes to dehydration unit. This unit contains a catalyst that helps to

remove the water from the gas. The water is removed is to avoid freezing the gas stream inside the pipe. The performance of the dehydration unit is monitored through a pressure transmitter, where the high pressure will result in changing the catalysis reaction inside the unit. After the water is removed, the gas will pass by the analyser to measure that there is no water through the pipe. An analyser is an instrument which conducts chemical analysis on samples or sample streams. When the gas has less than 1% water content, the system can accept it. The gas will leave the analyser at a temperature of 120°F passing through the pipe to the cooling unit, where it is cooled from 120 to -40 °F. The change in temperature is monitored through a temperature transmitter.

The Natural Gas Processing System architecture in Figure 51 shows where the Absorb Water service is used. Two Participants, the Dehydration Unit and Cooling Unit, have been specified, their interactions captured by the Absorb Water service contract. These two participants represent the roles that are connected to this service. Table 23 shows the SLA for the Absorb Water service. For instance, the Participants, Cooling Unit and Dehydration Unit, respectively represent the consumer and provider for the service.

Attribute	**Description**	**Source**
Name	Absorb Water	SoaML Service Contract Architecture for the natural gas processing system
Provider	The Dehydration Unit that provides the service	SoaML Participants Architecture for the natural gas processing system
Consumer	The Cooling Unit that uses the service	SoaML Participants Architecture for the natural gas processing system
Start time	The point at which the Dehydration Unit is requested to Absorb Water (e.g. 9:00:00 AM)	SoaML ServicesArchitecture for the natural gas processing system
Due time	The end time of the requested Absorption process (e.g. 9:00:30 AM)	SoaML ServicesArchitecture for the natural gas processing system
Duration	30 seconds	SoaML ServicesArchitecture for the natural gas processing system
(Pre-condition)	Receiving the requested information including the amount of water, percentage removal required, and	SoaML Service Contract Architecture for the natural gas

	brief details of the condition of the sweet gas.	processing system
(Post-condition)	Water absorbed and content less than 1%. Signal sent by transmitter at the expected arrival time	SoaML Service Contract Architecture for the natural gas processing system
Contribution to Safety	Overall severity is very severe (see SHA results)	SHA

Table 23: Absorb Water SLA

Table 24 shows an extract from the SHA results for the 'Absorb Water' service. The analysis establishes the potential safety criticality of the 'Absorb Water' service, based on the severity of the worst credible effects of the identified failure modes. As a result of the analysis, stringent safety requirements allocated are defined and allocated to the 'Absorb Water' service.

Service	**Failure Modes**	**Effects**	**Severity**	**SSRs**	**Mitigation**
Absorb water Context: *Absorption service means removing the water in gas dehydration processes*	Water not removed (i.e. water content is more than 1%)	Blockage in the pipeline; Damaged pipelines; Explosion	(Very Severe, 4)	The failure mode "water not removed" shall be prevented or mitigated.	Active monitoring and cross-checking between removing water and measuring sweet gas through the analyser. Also, checking the catalyst inside the dehydration drum to ensure it is within its service life e.g. needs to be replaced every 5 years.
	Water removed when not required	N/A	N/A	None.	None.
	Water removed later than intended	Freezing inside the pipeline; Blockage in the pipeline; Damaged pipelines; Explosion	(Very Severe, 4)	The failure mode "water removed later than intended" shall be prevented or mitigated.	Active monitoring of timing targets and strategies for recovery from timing-related failures through analyser.
	Water removed earlier than intended	N/A	N/A	None.	None.

	Inaccurate percentage of water removed (i.e. water content is less than 1%)	N/A No direct safety effects The system can handle that without any effect	N/A	None.	None.

Table 24: Absorb Water SHA

The Absorb Water service has two potential hazardous failure modes, namely:

1- Water not removed (i.e. water content is more than 1%);
2- Water removed later than intended.

In the SHA, the failure modes are intended to act as prompts for the analyst, and it may be necessary to consider more than one interpretation. For example, the first case, "water not removed" (i.e. water content is more than 1%), the service is not provided, which may lead to a large amount of water flooding into the pipeline toward the cooling unit. This water may then freeze inside the pipeline, which would block the pipeline and prevent the flow of gas. Ultimately, this could lead to damaged pipelines or an explosion. The level of severity is measured and classified based on a risk matrix (154). In the oil and gas domain, risk is commonly analysed and managed using three risk matrices. They are as follows:

- Consequence Matrix;
- Frequency Matrix; and
- Risk Matrix.

The consequence/severity matrix we use in this case study is depicted in Table 25 and is based on four types of potential consequence: worker safety, public safety, environment and economic (154).

	Worker Safety	**Public Safety**	**Environment**	**Economic (Annual)**
Low, 1	Reportable or equivalent.	None.	Limited impact that is readily corrected	$10.000 to $100.00

Moderate, 2	Hospitalization or lost-time injury.	Minor medical attention.	Report to agencies and take remedial action.	$100.000 to $1 million
Severe, 3	Single disabling.	Hospitalization or serious injury. Some local reporting.	Irreversible damage to low-quality land. Or clean-up of environmentally sensitive areas requires	$1 million to $ 10 million
Very severe, 4	Fatality or multiple serious injuries.	Fatality or multiple serious injuries. Massive negative publicity.	Months of clean-up work needed in environmentally sensitive areas	≥ $10 million

Table 25: Consequence Categories (154)

Once the consequences associated with an incident have been identified, the next step is to estimate the frequency with which the incident might occur. An example matrix is shown in Table 26.

	Frequency	**Comments**
Low, 1	< 1 in 1.000 years	Essentially impossible: " Once in a blue moon " or "meteor falling out of the sky"
Moderate, 2	1 in 100 years to 1 in 1.000 years	Conceivable – has never happened in the facility being analysed but has probably occurred in a similar facility somewhere else.
High, 3	1 in years to 1 in 100 years	Might happen once in an engineer's career
Very High, 4	> 1 in 10 years	It is likely that the event has occurred at the site if the facility is more than a few years old.

Table 26: Example of Frequency Matrix (154)

Table 27 shows the Risk Ranking Matrix that is used to categorise the level of severity/consequences and frequency. The colours should be interpreted as follows: red means very high risk, orange means high risk, yellow means medium risk and green means low risk.

	Consequence				
Frequency		**Low, 1**	**Moderate, 2**	**Severe, 3**	**Very severe, 4**
	Low, 1	D	D	C	C
	Moderate, 2	D	C	C	B
	High, 3	C	C	B	A
	Very High, 4	C	B	A	A

Table 27: Risk Ranking Matrix (154)

Table 28 explains the meaning of the four letters used in the Risk Ranking Matrix in Table 27 (154).

A (Red) Very High	*This Level of risk requires prompt action: money is no object, and doing nothing is not an option. An "A" risk is urgent. On an operation facility, management must implement Immediate Temporary Controls (ITC), while longer-term solutions are being investigated. If effective ITCs cannot be found, then the operation must be stopped. During the design phases of a project, immediate corrective action must be taken in response to an "A" finding, regardless of the impact on the schedule and budget.*
B (Orange) High	*Risk must be reduced, but there is time to conduct more detailed analyses and investigations. Remediation is expected within, say 90 days. If the resolution is expected to take longer than this, then an ITC must be put in place.*
C (Yellow) Moderate	*The risk is significant. However, cost considerations can be factored into the final action taken, as can normal scheduling constraints such as availability of spare parts or the timing of facility turnarounds. Resolution of the finding must occur within, say, 18 months. An ITC may or may not be required.*
D (Green) Low	*Requires action but is of low importance. In spite of their low-risk ranking, "D" level; risks must be resolved and recommendations implemented according to a schedule: they cannot be ignored. (Some companies do allow very low-risk ranked findings to be ignored on the grounds that they are within the bounds of acceptable risk.)* *If the hazard is associated with a change to an existing process, it is not always necessary to conduct a full risk ranking, particularly if the change does not make a fundamental alteration to the process itself. In these cases, it is enough just to check that the risk value does not shift from one square to another if it does not then no further evaluation is needed.*

Table 28: Interpretation of the Letters used in the Risk Ranking Matrix (151)

In the example of the "water not removed" failure condition, the likelihood of there being a problem with the catalyst is low. Nevertheless, the potential effects are very high. As a result, the level of impact is considered very severe. This is because the whole plant will need to be shut down in order to fix the damaged pipe or to explore ways to remove the freezing water.

Having identified the failure mode and assessed the effect and level of risk, we must prevent or mitigate the failure mode "water not removed". The mitigation used is active monitoring and cross-checking between removing water and measuring the flow of sweet gas through the analyser. Also, monitoring of the level of catalyst inside the dehydration drum is used, e.g. it needs to be checked every 5 years which is a typical service life for catalysis. This is identified from the design representation for the dehydration drum.

Another failure mode is when water is removed later than intended. This may lead to water passing within the pipeline, freezing inside the pipeline, blocking the pipeline and thus preventing the flow of gas (151). A level catalyst needs to be fitted with the level of water. Another possible case is damaged pipelines or explosion. The level of the impact considered is very severe. The mitigation used is active monitoring of timing targets and strategies for recovery from timing-related failures through the analyser. This is used in order to avoid any delay in the process flow.

Heat Exchange (Cooling) Service

The Heat Exchange service is provided by the Cooling Unit which reduces the gas temperature from 120°F to -40°F. If the temperature does not reach the intended temperature, it is clear that the gas is not passing to the Cooling Unit. This in turn indicates that a gas leak has happened in the pipe. On receiving an over-temperature indication, the system should generate an automatic signal to shut down the cooling unit.

Most heat exchangers operating in a gas plant are conventional shell-and-tube types and are ideal for steam and hot oil systems where fouling occurs. They are relatively inexpensive and easy to maintain because the tube bundle can be removed and tubes cleaned or replaced as needed. (150).

The Engineering Data Book (159) provides extensive details of the exchangers used in gas processing. Where the fluids are clean and fouling does not occur, such as in gas–gas exchangers, compact heat exchangers are ideal. Wadekar (2000) provides a good overview of the more common types.

In the event of gas leak case, the leak might occur at the beginning of the unit, in which case the control operator should notice a sharp increase in the temperature after the cooler indicates that no gas is passing through the cooler. The upstream emergency isolation valve should close automatically, since it has a logic where any sharp increase in temperature is detected and the valve closed in response. This is to avoid continuous gas leaking from the faulty location. The leaking gas is considered as a fuel and any spark of ignition will result in fire.

Here we apply SHA for the analysis of the Heat Exchange service. The SLA for the service is shown in Table 29 in which the Cooling Unit and the Customer respectively represent the consumer and the provider for the service.

Attribute	Description	Source
Name	Heat Exchange	SoaML Service Contract Architecture for the natural gas processing system
Provider	Cooling Unit that provides the service	SoaML Participants Architecture for the natural gas processing system
Consumer	The Customer that uses the service	SoaML Participants Architecture for the natural gas processing system
Start time	Point at which the request for cooling is sent to the Cooling Unit (1:00:00 PM)	SoaML ServicesArchitecture for the natural gas processing system
Due time	The end time of the gas cooling process (1:00:30 PM)	SoaML ServicesArchitecture for the natural gas processing system
Duration	30 seconds	SoaML ServicesArchitecture for the natural gas processing system
(Pre-condition)	The Cooling Unit has received the requested information including the amount of gas, percentage of cooling, and brief details of the condition of the gas situation	SoaML Service Contract Architecture for the natural gas processing system
(Post-condition)	Gas is cooled to the requested temperature and signal sent by the transmitter indicating the expected arrival time for Separation	SoaML Service Contract Architecture for the natural gas processing system
Contribution to Safety	Overall severity is very severe (see SHA results)	SHA

Table 29: Heat Exchange SLA

Table 30 shows an extract from the SHA results for the 'Heat Exchange' service. The analysis establishes the potential safety criticality of the 'Heat Exchange' service, based on the severity of the worst credible effects of the identified failure modes. As a result of the analysis, stringent safety requirements are defined for and allocated to the 'Heat Exchange' service.

Service	Failure Modes	Effects	Severity	SSRs	Mitigation
Heat exchange	Gas not cooling	Gas leak within pipelines	(Very Severe, 4)	The failure mode "gas not cooling" shall be prevented or mitigated.	Active monitoring and cross-checking the transmitter to control the liquid level in the

					cooling by establishing setting point in the system at -40 °F and pressure is 400 psig
	Gas cooling when not required	N/A	N/A	None.	None.
	Gas cooling later than intended	N/A	N/A	None.	None.
	Gas cooling earlier than intended	N/A	N/A	None.	None.
	Inaccurate percentage of gas cooling	Gas leak within pipelines since no gas is passing through cooler	(Very Severe, 4)	The failure mode "inaccurate percentage of gas cooling" shall be prevented or mitigated.	Early temperature cross-checking of the transmitter and confirmation that the setting point in the DCS that the cooling unit is -40 °F and pressure is 400 psig

Table 30: Heat Exchange SHA

The Heat Exchange service has two potentially hazardous failure modes:

1- Gas not cooling Water;
2- Inaccurate percentage of gas cooling.

The first failure mode may lead to a gas leak the within pipelines, since no gas is passing through the cooler. The level of impact of this failure mode is classified as Very Severe (4).

In this service, the transmitter plays an important role in setting the point in the Natural Gas Processing system at -40°F, which is essential to cool the gas. The proposed mitigation is active monitoring and crosschecking in order to deal proactively with this potentially dangerous failure mode.

Another failure mode is "inaccurate percentage of gas cooling" which may lead to a gas leak within the pipeline, since no gas is passing through the cooler. The potential frequency of this failure mode is assessed as High (3) and the impact is considered Very Severe (4). The risk matrix can help identify the resulting level of risk, which is identified as very high. The mitigation here is

early crosschecking of the transmitter and a confirmation of the setting point in the DCS. This can ensure that no more leaking gas goes into the atmosphere.

In the event of the above failure modes, the unit needs to be shut down to ensure that no more gas leaks to the atmosphere. In principle, the failure modes identified for the system described in this example could lead to the same problems as those encountered during the Bhopal Plant Disaster in 1984. At Bhopal, there were gas leaks and problems with the pressure gauge (160). The leak was running quite slowly and the pressure indicator instrument did not function as intended.

Separate Gas from Liquid

The Gas-Liquid Separation function is provided by the Cooling Unit. A Separator is designed to separate intermittent large volumes of liquids from a gas stream. It may take the form of vessels or a manifold pipe system. Figure 9 shows the basic structure of a vessel-based gas-liquid separation system.

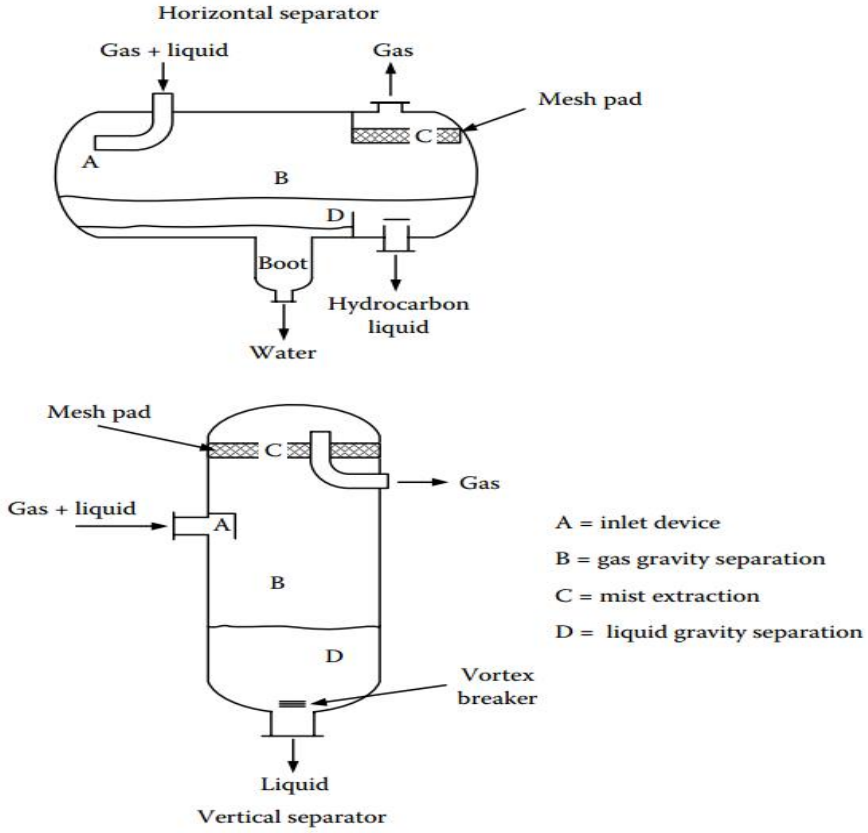


Figure 58: Gas-Liquid Separation, adapted from (159)

The separation vessel can be orientated vertically or horizontally. Vertical separators are usually used while gas flow rates are low or the liquid-to-gas ratio is low. They are preferred offshore as they occupy less space. Nevertheless, gas flow in vertical systems is upwards and opposes the flow of liquid droplets. Vertical separators are bigger and cost more than horizontal separators. The inlet suction scrubbers used by compressor stations are commonly vertical. Horizontal separators, on the other hand, are favoured for large volumes of liquid or in situations where the liquid-to-gas ratio is high. Lower gas flow rates and increased residence times can help provide better liquid dropout. The larger area offers better degassing and also more stable liquid levels. This is can be seen in Figure 58 (161).

After cooling, the gas is passed to the separation unit in order for the gas to be separated from the liquid. The drop in temperature helps condensed gas to liquefy. The liquid stream is called liquid gas. After separation, the liquid gas is sent to a Chemical Manufacturing Company while the gas goes to an Electricity Company.

We now apply SHA to the Separate Gas from Liquid service, defined in in the SLA shown in Table 31.

Attribute	**Description**	**Source**
Name	Separate gas from liquid	SoaML Service Contract Architecture for the natural gas processing system
Provider	Cooling Unit that provides the service	SoaML Participants Architecture for the natural gas processing system
Consumer	The Customer that uses the service	SoaML Participants Architecture for the natural gas processing system
Start time	Requesting Separation by the Cooling Unit (e.g. 11:00:00 AM)	SoaML ServicesArchitecture for the natural gas processing system
Due time	The end time of the gas-separation process (e.g. 11:00:30 AM)	SoaML ServicesArchitecture for the natural gas processing system
Duration	30 seconds	SoaML ServicesArchitecture for the natural gas processing system
(Pre-condition)	Receiving the request information including the amount of gas, percentage of separation, and brief	SoaML Service Contract Architecture for the natural gas processing system

	details of the condition of the gas situation.	
(Post-condition)	Gas is separated and send signal by transmitter and the expected arrival time to Customer	SoaML Service Contract Architecture for the natural gas processing system
Contribution to Safety	Overall severity is very severe (see SHA results)	SHA

Table 31: Separate gas from liquid SLA

Table 32 shows an extract from the SHA results for the 'Separate Gas from Liquid' service. In the example, the required separation level to be transmitted should be 80% for liquid gas and 20% for gas: a higher level than this may lead to wrong separation. There are two potentially dangerous failure modes, namely:

1- Gas not Separated;
2- Inaccurate percentage of gas separated.

The first failure mode is "gas not separated" which may lead to a mixing of the gases. The customer would receive a product does not correspond to the specification, which might have a negative impact on the customer's handling materials and lead to pipe rupture. The severity of this failure mode is assessed as Moderate (2). In the example, this failure mode is mitigated by monitoring that the level of liquid is maintained at 80%.

Service	**Failure Modes**	**Effects**	**Severity**	**SSRs**	**Mitigation**
Separate gas from liquid service	Gas not Separated	Damage to the equipment in customer plant	(Moderate, 2)	The failure mode "gas not separated" shall be prevented or mitigated.	Active monitoring and cross-checking by installing level transmitter at 80% to control the liquid level in the DCS
	Gas separated when not required	N/A	N/A	None.	None.
	Gas separated later than intended	N/A No direct safety effects	N/A	None.	None.
	Gas separated earlier than intended	N/A	N/A	None.	None.
	Inaccurate	Damage to the	(Moderate,	The failure	Early address

	percentage of gas separated	equipment in customer plant	2)	mode "inaccurate percentage of gas separated" shall be prevented or mitigated.	crosschecking transmitter level and confirmation of separation of the gas to avoid sending mixed gases.

Table 32: Separate Gas from Liquid SHA

Send Emergency Signal from Cooling Unit to Responder

In the event of an emergency, the control operator should call an emergency number, alerting the fire, hospital and security agencies to the situation. A response action should follow. Each of these agencies has a different responsibility in the emergency situation. Fire fighters are on continuous standby and should be ready to extinguish any fire that could result during a leak.

The ambulance team should provide the necessary medical support, since there is potential for people to inhale toxic gas, which could result in serious medical complications. The security team are responsible for emergency traffic control and also for ensuring that only authorised people can access the emergency location.

We have applied SHA to the 'Send Emergency Signal' service. The example below explains the use of SHA, considering the corresponding SLA as defined in Table 33.

Attribute	**Description**	**Source**
Name	Send Emergency Signal	SoaML Service Contract Architecture for the natural gas processing system
Provider	Responder that provides the service	SoaML Participants Architecture for the natural gas processing system
Consumer	Cooling Unit uses the service	SoaML Participants Architecture for the natural gas processing system
Start time	The point at which the Cooling Unit sends the emergency signal (11:00:00 PM)	SoaML ServicesArchitecture for the natural gas processing system
Due time	The end time of the Receive Emergency Signal request (11:00:10 PM)	SoaML ServicesArchitecture for the natural gas processing system

Duration	10 seconds	SoaML ServicesArchitecture for the natural gas processing system
(Pre-condition)	Send the signal including the current situation and location	SoaML Service Contract Architecture for the natural gas processing system
(Post-condition)	Receiving the signal	SoaML Service Contract Architecture for the natural gas processing system
Contribution to Safety	Overall severity is Very Severe	SHA

Table 33: Send Emergency Signal SLA

Table 34 shows an extract from the SHA results for the 'Send Emergency Signal' service. The analysis establishes the potential safety criticality of the 'Send Emergency Signal' service, based on the severity of the worst credible effects of the identified failure modes. As a result of the analysis, stringent safety requirements are developed and allocated to the 'Send Emergency Signal' service.

Service	**Failure Modes**	**Effects**	**Severity**	**SSRs**	**Mitigation**
Send emergency signal	Signal not sent	Damaged pipeline Explosion Fire	(Very Severe, 4)	The failure mode "signal not sent" shall be prevented or mitigated.	Use of redundancy by transmitter to confirm the signal has been received
	Signal sent when not required	N/A	N/A	None.	None.
	Signal sent later than intended	Damaged pipeline Explosion Fire	(Very Severe, 4)	The failure mode "signal sent later than intended" shall be prevented or mitigated.	Active monitoring of timing targets and strategies for recovery from timing-related failures.
	Signal sent earlier than intended	N/A	N/A	None.	None.
	Inaccurate signal sent (Wrong respond) (Wrong reading)	Damaged pipeline Explosion Fire	(Very Severe, 4)	The failure mode "inaccurate signal sent" shall be prevented or mitigated.	Early crosschecking by maintenance technician to confirm that correct the signal has been received.

Table 34: Send Emergency Signal SHA

The first failure mode is 'signal not sent' which may lead to a damaged pipeline, an explosion or a fire. The level of impact is assessed as Very Severe (4). In the example, the suggested mitigation for this failure mode is the use of redundancy by the transmitter to confirm that the signal has been received.

6.4.6.2. SFA

In this section, we show how SFA is applied to two primary flows in the BPMN model for the Natural Gas Processing systems, which are as follows:

- Flow between 'Send emergency signal' (Cooling Unit) to 'Receive signal and check the status' (Responder);
- Flow between 'Exchange heat from 120°F to -40°F at pressure 400 psig' to 'Monitor that the temperature is at -40 °F, the pressure is at 400 psig and send signal by transmitter'.

Flow from 'Send Emergency Signal' to 'Receive Signal and Check the Status'

The performance of the Cooling Unit is monitored to detect any gas leakage: the control operator should notice a sharp increase in temperature after the cooler indicating that no gas is passing through the cooler. The upstream emergency isolation valve should close automatically. This is to avoid continuous gas leaking from the leak location.

The SFA here considers 4 main types of deviation, suggested by the following guidewords (Table 35):

- **Omission;**
- **Commission;**
- **Late;**
- **Value.**

Omission may be total or partial. Total omission is when the signal is not communicated at all. This may be due to a conversion fault (Incorrect input or server crashed), which could result in the lack of an emergency request signal. The mitigation suggested here is redundancy in sending sources to ensure accurate communication. Partial omission, in which part of a communication is

missing, may be due to the required resource being missing or service/server incompatibility, which could result in an inconsistent signal. The proposed mitigation is to ensure accurate communication through checking and monitoring API functions to describe all the valid messages that one program can accept. Another example is related to commission failure modes (repetition or spurious communication). Repetition of the signal with additional output may lead to a redundant description (due to incorrect input or server crashed), which could result in a wrong response, wrong reading of the signal and receipt of a conflicting request. The proposed mitigation is to ensure safe communication through the use of high-integrity components between participants to detect incorrect signals.

ID (Flow)	Guide Word	Deviation	Possible Causes	Effects	DSSRs	Mitigation
1.1	Omission (no communication within the signal)	No output signal is sent	Conversion fault (Incorrect input) (server crashed)	No request received	The failure mode “no output signal is sent” shall be prevented or mitigated.	Use of redundancy in sending sources
	Omission (Incomplete signal)	Part of output signal is missing	Required resource missing Service/server incompatible	Inconsistent signal	The failure mode “part of output signal is missing” shall be prevented or mitigated.	Checking and monitoring APIs function and fault containment
1.2	Commission (Repetition)	Additional output signal sent	Redundant description (Incorrect input) (Server crashed)	Conflicting request received Wrong response Wrong reading signal	The failure mode “additional output signal sent” shall be prevented or mitigated.	Use of high-integrity components between participants to detect incorrect signal with sufficient reliability and accuracy
	Commission (Spurious)	The communication is unintended	Conversion fault (Incorrect input) (server crashed)	Only affected when the message is valid	The failure mode “the communication is unintended” shall be prevented or mitigated.	Use of high-integrity components between participants to detect incorrect signal with sufficient reliability and accuracy
1.3	Early	N/A	N/A	N/A	N/A	Just: the output cannot be produced
1.4	Late	Unacceptable response time to change in demands	Execution fault Time out (severe crash or communication failure)	Overloading signal	The failure mode “Unacceptable response time to change in demands” shall be prevented or mitigated.	Active monitoring by transmitter of timing targets and strategies for recovery from timing-related failures

1.5	Value	One component of output is not proportional to demand	Incompatible components (Composition fault)	False signal Hard to detect from flow of wrong advice through radio communication channel as well	The failure mode "One component of output is not proportional to demand" shall be prevented or mitigated.	Applying diversity in the flow and fault containment to detect and reject incorrect signal

Table 35: 'Send Emergency Signal from Cooling Unit' to 'Receive Signal and Check the Status' SFA

In the case of spurious communication, the communication is unintended which may lead to a conversion fault (Incorrect input or server crashed). The proposed mitigation is again to ensure adequate communication through the use of high-integrity components between the participants. If the flow occurs later than intended, this may lead to unacceptable response times, to changes in demand and to a 'failure in execution fault' or time out (severe crash or communication failure). The potential effect is to overload the signal and the proposed mitigation is to provide active monitoring by the transmitter of timing targets and strategies for recovery from timing-related failures. The potential deviation uncovered by application of the guide word 'value' is that one component of output is not proportional to the demand received due to incompatible components (composition fault). The effect is transmission of a false signal, which is hard to detect from the flow resulting in the wrong advice through the radio communication channel. The proposed mitigation is to ensure accurate communication by applying diversity in the flow and fault containment strategies to detect and reject incorrect signals.

Flow from 'Exchange Heat from 120°F to -40°F at Pressure 400 psig' to 'Monitor that the Temperature is at -40 °F, the Pressure is 400 psig and Send Signal by Transmitter'

The SFA here considers 3 main types of deviation, suggested by the following guidewords (Table 36):

- **Omission;**
- **Commission;**
- **Value.**

One important aspect of the analysis generated from SFA in this case study is the commonality in the use of the types of mitigation, which should help to simplify the detailed design by focusing on a limited set of high-quality mitigation measures (mainly redundancy and reliance on high-integrity components) that can potentially address the identified failure modes.

ID	Guide Word	Deviation	Possible Causes	Effects	DSSRs	Justification/Design Recommendations
1.1	Omission (no communication within the flow)	No output process is sent	conversion fault (Incorrect input) (server crashed)	Request not received	The failure mode "no output process is sent" shall be prevented or mitigated.	Use of redundancy in sending sources
	Omission (Incomplete communication)	Part of output signal is missing	Required resource missing Service/server incompatible	Inconsistent temperature	The failure mode "part of output signal is missing" shall be prevented or mitigated.	Checking and monitoring APIs function and fault containment
1.2	Commission (Repetition)	Additional output process sent.	Description incorrect or mismatching Execution fault (Incorrect result)	Unwanted change Only effected when the flow is valid	The failure mode "Additional output process sent. No change is required" shall be prevented or mitigated.	Use of high-integrity components between participants to detect incorrect signal temperature indications
	Commission (Spurious)	The communication is unintended	Conversation fault (Incorrect input) (server crashed) Accounting problems	Only effected when the message is valid	The failure mode "the communication is unintended" shall be prevented or mitigated.	Use high-integrity components among participants to ensure the message received is exactly the same as the message that was sent
1.3	Early	N/A	N/A	N/A		Just: the output cannot be produced
1.4	Late	N/A	N/A	N/A		N/A

1.5	Value	One component of output is not proportional to demand	Undetected failure in any process. Corruption message (Execution fault incorrect result- or servicer crash)	False temperate Can be detected from flow (Gas leak) or temperature degree	The failure mode "One component of output is not proportional to demand" shall be prevented or mitigated.	Applying diversity in the flow and early cross-checking and fault containment

Table 36: 'Exchange Heat from 120 to -40 °F at Pressure 400 psig' to 'Monitor that the Temperature is at -40 °F, the Pressure is 400 psig and Send Signal by Transmitter'

6.4.7. Safety Cases

This section presents the results of applying the SOA safety argument pattern catalogue defined in Chapter 5 to the Natural Gas Processing case study introduced in this Chapter. The purpose of this part of the case study is to examine how the SOA safety argument pattern catalogue can help in communicating the justification for the safety of the Natural Gas Processing system and its associated services. It also shows how the analysis process, as applied to the case study, relates to the overall structure of the SOA safety argument. The case study will also be used to evaluate how the safety requirements and mitigation arising from the analysis can be used to define the safety argument contracts based on the SLAs defined during the design.

6.4.7.1. Gas Service System Safety Argument

The safety case for the Gas Service System starts with an instantiation of the Top Argument pattern. The Top Argument includes a services-directed argument, which covers the main safety claims concerning the Natural Gas Processing System. The results of the instantiation are shown Figure 59

G_SOA presents the claim for the overall safety of the Natural Gas Processing System in the context of the use case, i.e. the list of services covered (C_SOA). The strategy decomposes the argument over all of the identified services within the system, based on the SOA specification defined in SoaML. For each service, the argument makes a claim that the service is acceptably safe. This is done by means of creating an away goal for each service , which references the corresponding Service Argument module. In this section, we only focus on the argument for the 'Send Emergency Signal' service. This service is enacted in the event of a gas leak occurring at the beginning of the Cooling unit; the control operator is expected to detect a sharp increase in temperature based on the cooler indicating that no gas is passing through. The upstream emergency isolation valve should close automatically. The leaking gas is a fuel and any spark of ignition will result in fire. Additional safety arguments are described in Appendix E.

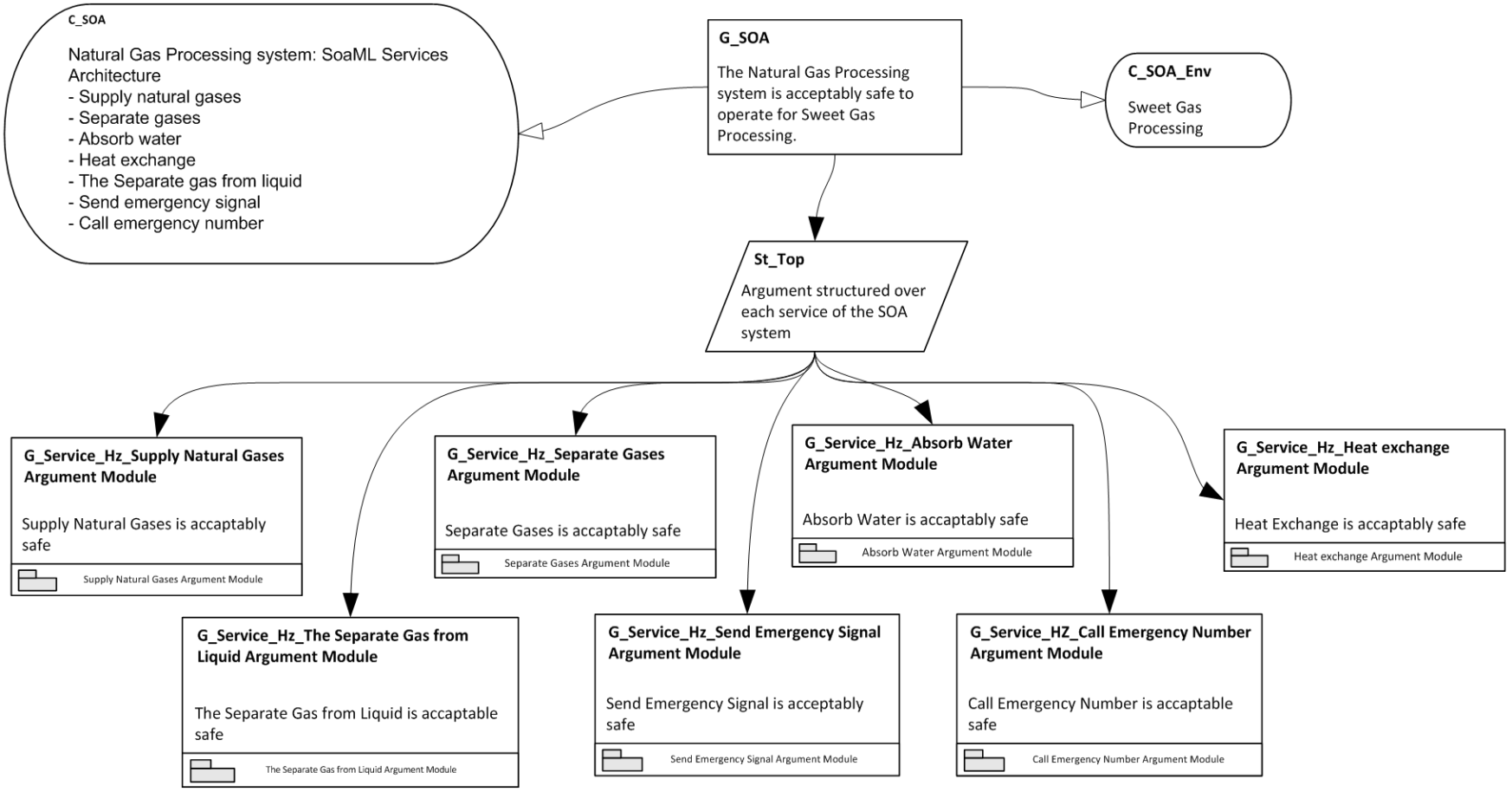


Figure 59: Top Argument Module

6.4.7.2. Service Argument for Send Emergency Signal

The argument structure in Figure 60 shows the instantiation of the Service Argument pattern for the 'Send Emergency Signal' service. The instantiation is based on the result of the SHA analysis of the 'Send Emergency Signal' service presented in section 6.4.6.1. The top-level claim for the Send Emergency Signal states that the service is acceptably safe, in the context of the SOA specification defined in SoaML CollaborationUse. The strategy argues over all the identified service hazards within the service based on the analysis carried out in the SHA. This is made in the context of the results of the modelling and SHA for this service.

The argument focuses on three failure modes, which have hazardous effects, which have been generated from the SHA results, namely:

- Signal not sent;
- Signal sent later than intended;
- Inaccurate signal sent.

The argumentation strategy adopted for each failure mode is to argue that the failure mode has been mitigated to the required level of integrity. For example, the failure mode 'Signal not sent' is mitigated by use of redundancy in the transmitter to confirm that the signal has been received. The mitigation claim is then defined in the form of a safety contract post-condition that is supported by the SLA Contract Argument.

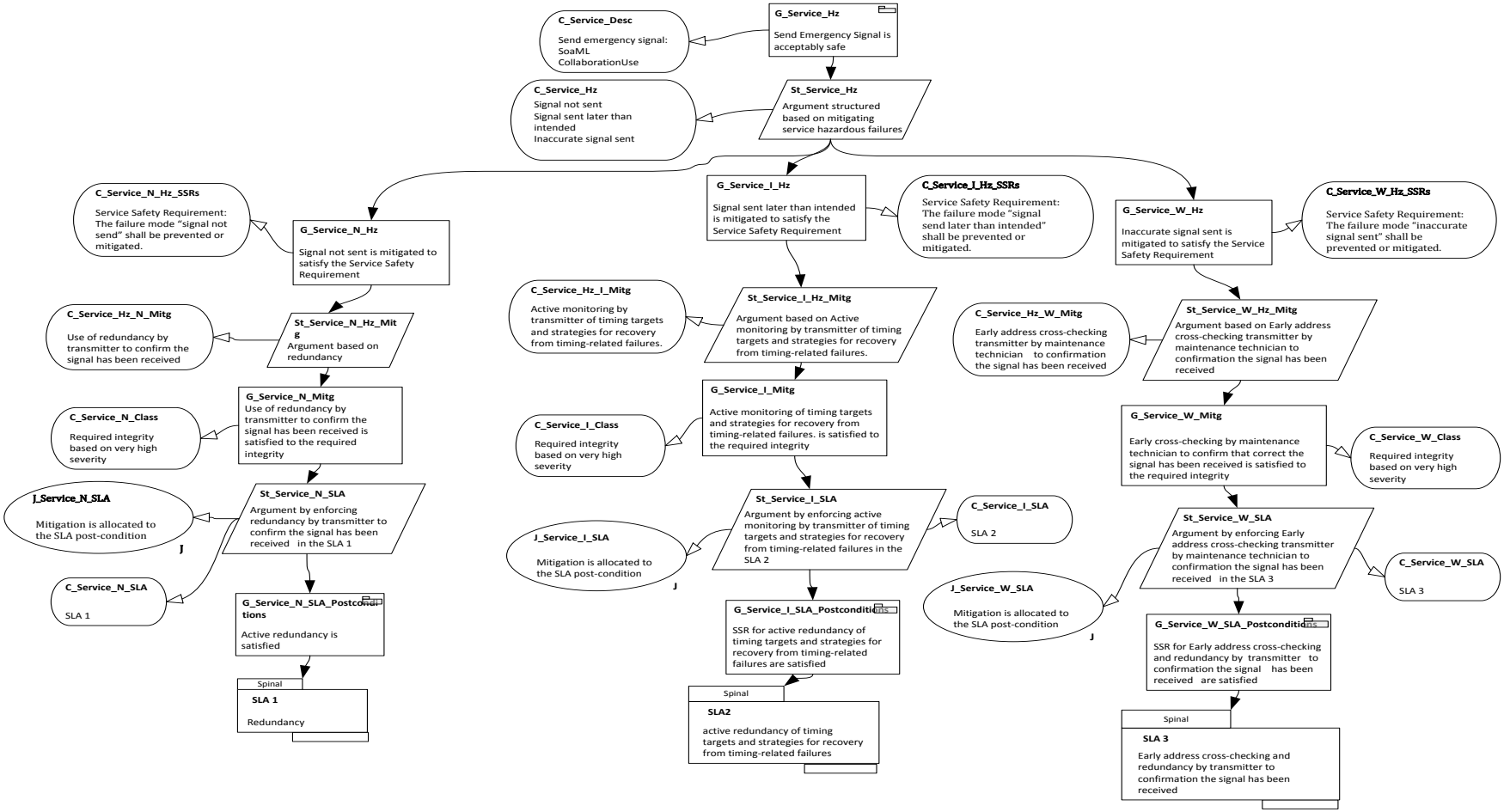


Figure 60: Service Argument for 'Send Emergency Signal' Service

6.4.7.3. SLA Contact Argument for Send Emergency Signal Service

Figure 61 shows the instantiation of the SLA Contract Argument pattern for the 'Send Emergency Signal' service. The top-level away goal states that the mitigation requirement for active redundancy is satisfied. This corresponds to the post-conditions of the SLA associated with the 'Send Emergency Signal' service. The argument strategy is to argue over the satisfaction of the set of SLA pre-conditions. This takes the form of away goals that references claims stated in the Participant Arguments for the participants satisfying the SLA pre-conditions. For example, the away goal which points to the Cooling Unit argument module references a claim in that module which states that the Cooling Unit sends a signal to the Responder and monitors to check that the signal has been received within 60 second. If the Responder does not receive it, the Cooling Unit backup should send another signal to the Responder after 60 seconds.

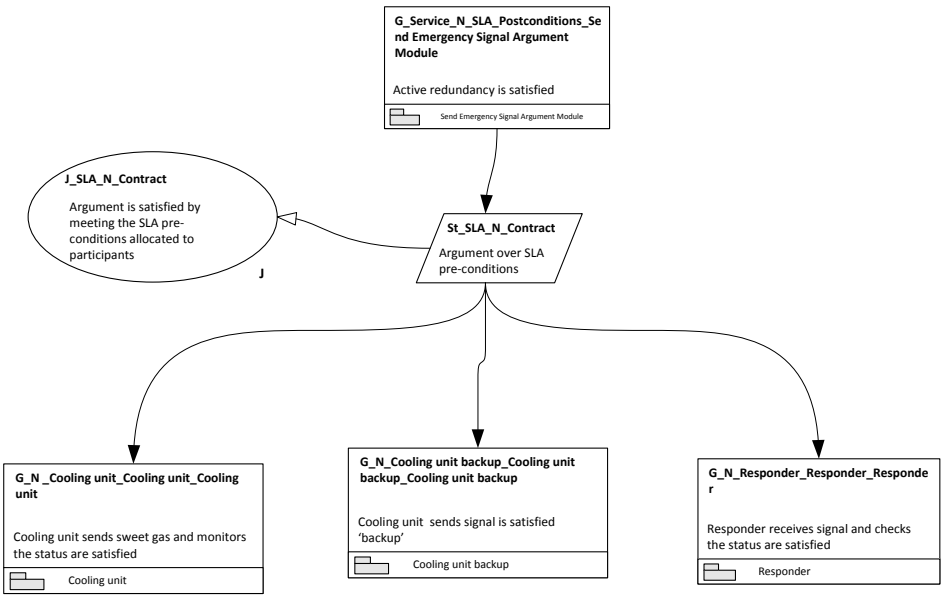


Figure 61: SLA contact Argument for 'Send Emergency Signal' Service

6.4.7.4. Participant Argument for the Cooling Unit Participant

Figure 62 shows the instantiation of the Participant Argument pattern for the Cooling Unit Participant. The argument is based on the satisfaction of the pre-conditions allocated to the Cooling unit Participant via the SLA in which the Participant is involved; namely, that the Cooling Unit sends a signal to the Responder. The argument then provides a justification for how the above claims are supported based on the results of the SFA and modelling in BPMN.

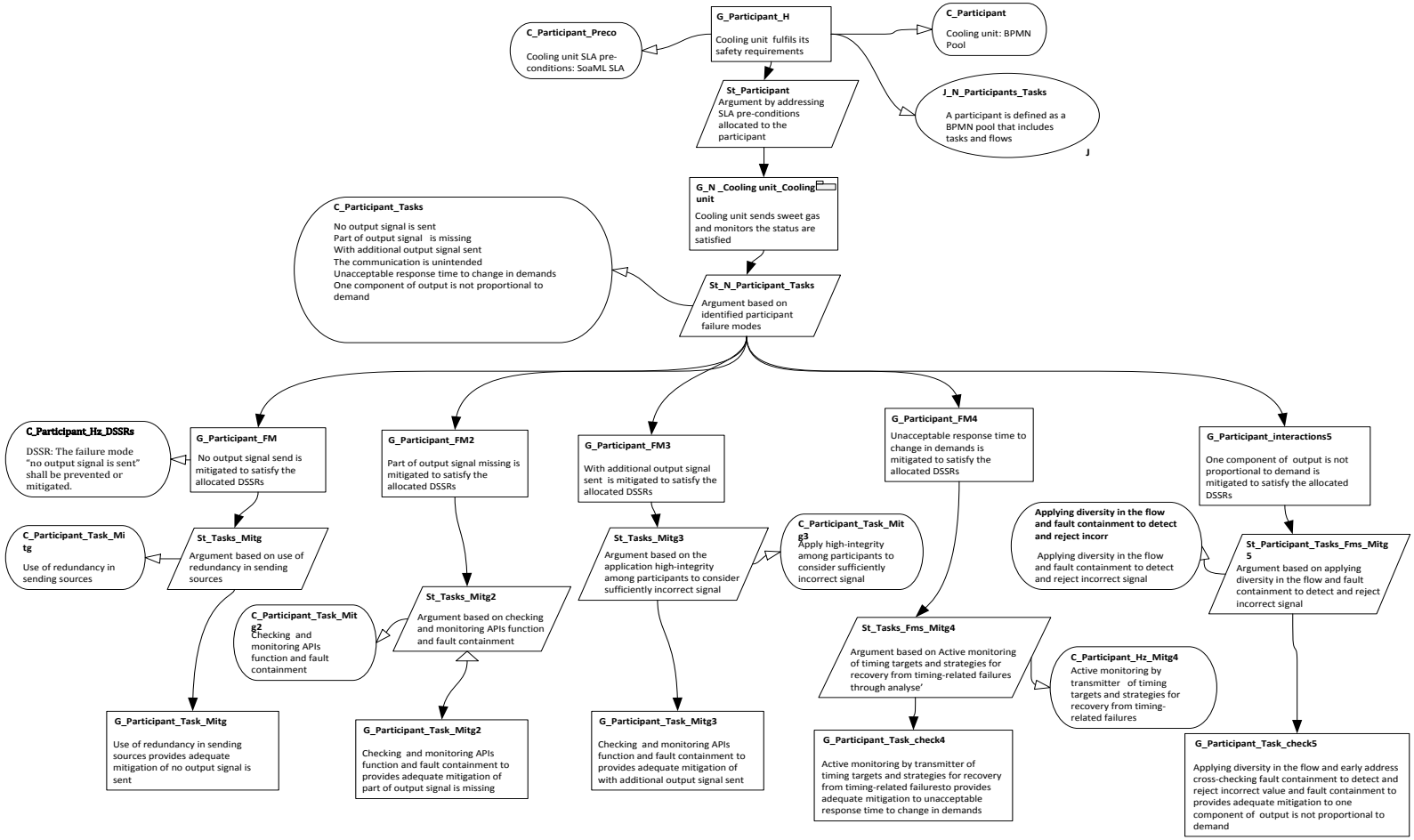


Figure 62: Participant Argument for the Cooling Unit

6.5. Questionnaire-based Evaluation

In order to provide an independent evaluation of our approach, we conducted semi-structured interviews with two engineers[1] from the Oil and Gas industry and one senior safety engineer from the healthcare domain.

The interviews started with an explanation of the interview protocol using the forms shown in Appendix F, which highlight the voluntary nature of the exercise and the fact that the confidentiality of the participants will be maintained. The presentations recorded in Appendix F were used to explain the background to and the main contributions of the research and each of the case studies respectively.

This was followed by a set of open questions addressing the three key contributions of this thesis. The questions are shown in Table 37.

Date: Areas of expertise of the interviewee:
1- What do you think about the overall approach?
2- Do you think that the modelling approach improve the understanding of the system?
a. What aspects in particular did you find useful?
b. What aspects in particular did you find hard to use?
3- Do you think that the safety analysis methods, SHA and SFA, can provide a systematic approach to analysing services?
a. What aspects do you think have the potential to improve practice?
b. What aspects do you think are infeasible to use?
4- Do you think that the safety case used improve the justification for the safety of the service-based system?
a. What aspects do you think have the potential to improve practice?
b. What aspects do you think are infeasible to use?
5- Do you think that the modelling, safety analysis and safety case artefacts are clearly integrated?
6- Can you provide suggestions for areas that can improve the framework?

Table 37: Interview Questions

The answers to the interview questions are listed in Appendix F. Below is our analysis of the outcome of the interviews.

[1] Please note that one of these engineers was involved in the development of the Natural Gas Processing Case Study. Nevertheless, the views of this participant concerning the evaluation of the proposed approach, though not fully independent, provided considerable insights concerning the effectiveness of the approach.

Question 1: What do you think about the overall approach?

All of the participants agreed that the proposed approach was useful for their particular domain and for service-based systems. One participant mentioned that it could provide a strong basis for risk assessment.

Question 2: Do you think that the modelling approach improves the understanding of the system?

All of the participants agreed that the modelling approach improved the clarity of the system description. One participant highlighted the fact that focusing on services rather than functions was useful, particularly for non-experts. They all emphasised the benefits of the separation between structure and behaviour enforced by the approach.

Question 2a: What aspects in particular did you find useful? Ease of use was a feature of interest to the participants. Representing the system graphically was also mentioned as a useful aspect by one of the participants.

Question 2b: What aspects in particular did you find hard to use? The process of generating the models is manual, and two engineers highlighted the need for additional automated modelling functions. One engineer emphasised the need for initial training (mainly for non-engineers).

Question 3: Do you think that the safety analysis methods, SHA and SFA, can provide a systematic approach to analysing services?

All participants agreed on the benefits of the systematic nature of the analysis, highlighting the clear step-by-step process. One participant highlighted the benefits of adapting existing safety analysis techniques, which should already be familiar to many safety engineers.

Question 3a: What aspects do you think have the potential to improve practice?

The clear integration between analysis and modelling was mentioned in addition to the potentially repeatable nature of the analysis.

Question 3b: What aspects do you think are infeasible to use? This was a hard question for the participants to answer, since they did not have experience of fully applying the approach in practice. One participant highlighted possible language inconsistencies, e.g. mapping between what some call 'fault' and others 'failure' (though this is a problem which can be observed in many safety analysis techniques).

Question 4: Do you think that the safety case used in the approach improves the justification of the safety of the service-based system?

Having a safety case pattern was highlighted as a useful feature. The modular nature of the justification was mentioned as a positive aspect, particularly in showing the big picture for the safety argumentation.

Question 4a: What aspects do you think have the potential to improve practice?

The explicit link between the analysis and the safety case was highlighted as an advantage, showing how the safety case is built on the results of SHA and SFA and therefore improving traceability.

Question 4b: What aspects do you think are infeasible to use? Again, this was a hard question to answer as the engineers did not apply the framework in a real world context. However, one participant was sceptical of the benefits of, or the need for, the detailed arguments and how they are going to be perceived in terms of effort.

Question 5: Do you think that the modelling, safety analysis and safety case artefacts are clearly integrated?

All of the participants agreed that the integration between the different elements of the approach was clear.

Question 6: Can you provide suggestions for areas that can improve the framework?

Two participants were keen that the model-based weaving approach be implemented (an area for future work in this thesis as discussed in Chapter 7). One engineer highlighted the importance of addressing a practical concern:

how much details can be considered enough in terms of depth, e.g. can we stop at the SLA level, abstracting the details about participants?

6.6. Evaluation of Thesis Contributions

This thesis focuses on four contributions made during the development of this research, namely:

- SOA Modelling;
- SOA Safety Analysis;
- SOA Safety Cases;
- Tool Support.

Each of these contributions has been evaluated and the results of this evaluation are discussed in the next four sections. Safety is a qualitative attribute and as such the results generated by this research, in terms of "improving safety analysis" are qualitative in nature, mostly based on our two qualitative case studies.

6.6.1. SOA Modelling

During this research, we developed design models of two case study systems based on information directly generated from the domain of interest, specifically healthcare and oil and gas. Domain knowledge is key in high-risk industries, since safety is a domain- and context-specific property. Evidence shows that a poor understanding of the domain of the system is a major contributor to safety accidents and incidents, especially for computer-based systems (162).

Here, the main contribution of this research is in showing how SoaML and BPMN can be used systematically in the design and assurance of safety-critical SOA. Based on feedback from the case studies, the following evaluation observations can be made:

- **SoaML and BPMN representation is considered important to understanding the safety of resulting design**: Since SoaML represents the system structure and BPMN represents the system behaviour: modelling both of these aspects using structured modelling languages

has helped in providing a holistic understanding of the design of the system. This reflects the fact that hazards can arise from the system structure as well as its behaviour.

- **Integrated modelling and analysis offers greater transparency**: The way in which the SOA system models and analyses are developed provides an integrated approach to modelling and a clearer basis for safety analysis. This means that the results of the analysis can clearly show which parts of the system could potentially have contributed to hazardous failures.
- **Traceable integration between system design and safety analysis:** Building on the previous point concerning integration, the clearly defined architecture of the proposed SOA system, based on SoaML and BPMN, forms the basis for a model-based integration between the design and safety analysis. This model-based integration improves traceability and supports maintenance of the safety case and the safety analyses when they are generated in SOA development.

One drawback of our work is the lack of coverage of all potential SoaML design diagram types (e.g. capabilities) which might reveal further contributions to safety. These additional SoaML diagrams could have helped us provide more detailed safety analyses, e.g. of lower-level system failures. The reason for this was that it was not within the scope of our work, which mainly focused on safety at the requirements and architecture levels. Another limitation is that we have not implemented all the features of BPMN modelling, particularly model execution and simulation, which could have been useful in showing even more sources of hazardous behaviours or more detailed execution failure modes.

6.6.2. SOA Safety Analysis

During the analysis stage, we performed multiple analyses of the systems under consideration using the SHA and SFA methods we developed. The following evaluation observations can be made:

- **Systematic consideration of safety analysis:** SHA and SFA provided a systematic means for examination of the hazards posed by services and

for understanding how the detailed failure behaviour of these services can contribute to hazards. These safety analysis methods provided step-by-step processes for carrying out the SOA analysis. SHA and SFA extended FFA and SHARD by adding additional guidance, steps and data so that the results could be applicable and compatible with SOA specific characteristics (e.g. the notion of services, participants and tasks). As indicated by the interviews, practitioners found the focus on services and the associated SHA results more focused than the use of the conventional and potentially generic HAZOP technique.

- **SHA and SFA offer explicit means for generating service safety requirements:** This is one of the most important outputs of any safety analysis technique. Safety requirements are vital in driving the design by clearly specifying what criteria need to be taken into account by the architects so that the identified service hazards and detailed failure modes are eliminated or mitigated. This will help realise the demands of best practice for achieving assurance-based development (163).

One drawback of our analysis methods is the potential generation of too many SHA and SFA tables with little automated support. For example, for each service and flow we had to create a detailed analysis table and we ended up with a large volume of data: this was a labour intensive activity. Another limitation of this work is the limited integration between failure modes and the recommended types of suitable design tactics. This is an area where predefined rules connecting the types of failure modes and the categories of tactics could be defined in advance in order to provide proactive recommendations for the architects of safety-critical SOA, building on current best practice in SOA design and the published literature.

6.6.3. SOA Safety Cases

Safety cases have become a fundamental requirement in many safety-critical domains. Nevertheless, there is still very little research on how they can be applied to services and SOA. In this research, we have provided a systematic approach to generating a safety case for an SOA in a traceable and clear manner. The following evaluation observations can be made:

- **SOA safety case patterns:** This research developed safety case patterns for justifying the design of SOA which can be reused by the architects and analysts of services in isolation of any particular domain. In our case, we focused on healthcare and oil and gas. However, these safety case patterns are domain-independent and are not limited to these two domains.
- **Safety case traceability**: Safety case patterns were traceable, at the model-level, to the sources of information (i.e. in the SoaML, BPMN, SHA and SFA models). This has helped in showing how each feature of the design is justified in the safety case based on the existing design descriptions and analysis evidence.
- **Safety case structure mirroring the system design:** This mirroring has improved the representation of system safety reasoning by clarifying the relationship between the design and safety evidence. By looking at the organisation of the argument and the structure of the system design, we can more clearly understand the correspondence between them.
- **Separation of concerns:** This is enforced via argument contracts based on SLAs (i.e. concerns about hazards versus concerns about failures). The SLA helps us create a separation between the justification of the mitigation of the hazards which are important at the system level and the design failure modes that can contribute to the these hazards at a more detailed level.

One limitation of this work is the lack of automated support for argument pattern instantiation from models and analyses. Although argument pattern instantiation is predominantly manual, adding automated capabilities would improve the efficiency of the process. In our work, we provide a preliminary basis for weaving the argument patterns into the source design and analysis models and metamodels (please see the preliminary weaving model in Appendix G). However, this approach was not fully evaluated in the case studies. Further work here includes the evaluation of this automated approach via the case studies.

6.6.4. Tool Support

During this research, tooling and automated support was developed for modelling and safety analysis for the SOA environment. This tooling comprises the integrated Eclipse tool plugins that have been developed to support our SOA safety assurance framework. The following evaluation observations can be made:

- **Implementing model-based integration of SOA system structural and behavioural models:** The integration of models in this research offers greater transparency about how the models are developed and how they relate to one another. The overlap provides a specific emphasis on the relation between the structure of a service and its behaviour. In the tooling, users can automatically navigate between the different views of the SOA design, e.g. selecting a service and being able to review the corresponding low-level behaviour (detailed tasks and flows).
- **Automated support for traceability:** The use of SoaML and BPMN allows services and tasks to be based on machine-checkable models. This is implemented in a way that generates traceability links from SoaML models to BPMN models to indicate how the BPMN process implements the SoaML services and vice versa. This also provides the ability to apply the safety analysis in a more interactive manner.
- **Automated means for supporting SOA safety analysis:** The tool-support is extended to provide semi-automated configuration of the system safety analysis methods. The extension of the modelling environment is implemented to support SHA and SFA results by providing automated means for capturing and displaying the safety analysis for the services and the flow between tasks. This means that the results of the analysis can clearly show which parts of the system could have potentially contributed to hazardous failures.
- **Automated means for supporting SLA:** The tool also provides semi-automated configurations of SLAs between a certain service

provider and its consumers. This creates and integrates SLAs into the SoaML service contracts for the representation of the SLAs.

A limitation of this work, as mentioned earlier, is the lack of automated support for argument pattern instantiation from SOA models and SHA and SFA results. Partly, this is due to the use of different tool plugins, which do not offer clear interfaces that support interoperability. These tools were useful for demonstrating a proof of concept but could not easily be extended for a complete and fully functional toolset. Revisiting the choice of tools and plugins is an area for further work.

6.7. Hypothesis Revisited

The hypothesis made in this thesis is as follows:

> **Integrated** architectural modelling and analysis of safety-critical service-oriented systems provides a **traceable**, **consistent** and **systematic** means for **assuring** the safety of these systems

In this section, we reflect on the results of the evaluation with respect to the main terms used in the hypothesis.

6.7.1. Integration

Achieving integrated design and safety assurance is a key aim within the safety domain. In this thesis, we address this aim for SOA engineering. The activities for modelling, safety analysis and safety cases for SOA are integrated both at the level of the methodology (e.g. a clear model in SoaML is a prerequisite for instantiating the SOA patterns for safety cases) and also at the model level (e.g. model features in SoaML and BPMN are linked to model elements, represented in a tabular format, in the SHA and SFA). As a result, the SOA Safety Assurance Framework provides a clear and structured basis for integrated design, safety analysis and safety case development with explicit considerations of hazardous behaviour and safety justification.

6.7.2. Traceability

In this thesis, forward and backward model-based traceability between the design and safety analysis provided a seamless means of understanding and

checking the relationship between the SOA structure and behaviour and their safety consequences. Ensuring traceability is a fundamental requirement for all safety standards. The value of the safety analysis can easily be threated if its relation to design is not well understood. During this research, this thesis also considered a new way for linking the argument pattern instantiation with the design modelling and analysis processes through a weaving model (164). The weaving model in Appendix G provides a preliminary basis for automating argument pattern instantiation based on the source design and analysis models and metamodels.

6.7.3. Consistency

In the SOA Safety Assurance Framework, design features are reflected in the safety analysis and assurance and vice versa. This helps assess the completeness of how the different design features are covered by the safety analysis and safety cases. Inconsistencies can either be detected through the tool-supported traceability techniques implemented in this research or through pattern instantiation, e.g. by assessing claims that cannot be instantiated due to incomplete design structures or contracts or incomplete safety analysis results.

6.7.4. Systematicness

One of the weaknesses identified in the current literature concerning SOAs and safety-critical systems was the lack of a systematic method for assuring SOAs. One of the aims of this thesis was to provide a coherent step by-step process, by applying and adapting techniques available within SOA and safety domains. For each step within the framework, it is clear which methods are being used and which design models of are being analysed. This thesis uses the SOA concept in order to provide the necessary modelling languages with clear modelling steps. Next, the analysis is focussed on analysing individual services and their associated interactions in order to identify hazardous behaviours associated with the system, based on two well-defined processes (i.e. SHA and SFA). This is followed by the definition of a set of reusable safety argument patterns for generating the SOA safety case.

To illustrate how the framework may be applied in practice, the process was applied to two realistic case studies. The results of the case studies were validated by domain experts, who considered the approach to be potentially helpful in the safety analysis processes used in their own systems and environments, particularly due to the systematic and clear basis of the assurance framework. One of the key features of the framework is that the rationale for the approach is clear: each stage is performed for a specific reason producing very concrete artefacts, e.g. design models, analysis results or specific safety arguments.

6.7.5. Assurance

The SOA safety argument pattern catalogue created in this research follows a reasoning chain from the Top Argument until the Participant Argument. It can be instantiated for justification of the design and analysis for generating trustworthy evidence regarding the safety of deploying an SOA-based system within a specific safety application. These patterns are directly related to each other. The SOA argument pattern catalogue includes rules for pattern composition and traceability which are linked to the architectural design (i.e. SoaML and BPMN models) and analyses (i.e. SHA and SFA). These patterns were useful in showing how the results of modelling and analysis can support the development of explicit SOA arguments.

6.8. Threats to Validity

In this section, we describe the main threats to the validity of the findings of this thesis.

Firstly, although our modelling approach is biased towards the use of SoaML and BPMN, the overall framework does not depend on the specific use of these notations, as long as well-defined modelling languages are used for capturing the structural and behavioural aspects of an SOA design. This can be seen as a threat to external validity. However, for the sake of focused evaluation, it was important to limit the modelling to two specific notations.

Secondly, the choice of the domains for evaluation was important. We chose two different domains for the two case studies in order to evaluate the

capability of our framework against diverse challenges. Oil and Gas and Healthcare are two conceptually different domains. The first has well-established and detailed safety practices while the second only has high-level safety processes, particularly for computer-based systems (see Section 6.3.1 for a more detailed discussion). Due to the scale of the work, it was difficult to apply our approach to all key safety-critical industries. However, the two case studies cover the two main categories of domains in safety engineering, i.e. traditional (Oil and Gas) and emerging (Healthcare).

Thirdly, there is a great interdependency between the modelling, safety analysis and safety cases within the framework. This might affect the scalability and wider applicability of the framework if any of the underlying notations and techniques used are changed. That is if a different safety analysis techniques was used it may not scale as well as the one which was used e.g. the use of FTA rather than SFA. This is a limitation from a scalability point of view. However, there is a tradeoff between the wider usefulness of the approach and ensuring internal consistency between the different components of the framework.

Fourthly, the sample in the interviews is small (three interviewees). The experts hold senior roles and have been involved in the engineering of large-scale safety-critical services. Safety-critical information is hard to collect due to the sensitivity of the domain. Therefore it was decided that one-one interviews with experienced practitioners would allow us to evaluate the general validity of the approach and the potential for scalability for use in actual industrial contexts.

Finally, the proposed approach was implemented in the form of research prototype tools. Although they were useful in the case studies, the effectiveness and reliability of these tools for other uses cannot be demonstrated.

6.9. Chapter Summary

This chapter presented the results of the evaluation of the four main contributions of the research presented in this thesis. It revisited the research hypothesis and explained the means for evaluation used in this research. This evaluation was carried out by means of case studies, semi-structured

interviews and peer review. These forms of evaluation produced results supporting the hypothesis of the thesis which also highlighted potential limitations and possible areas for future work. The final conclusions of the research are presented in the next chapter.

Chapter 7

7. Conclusions

This chapter provides a summary of the main contributions of the research presented in this thesis and proposes areas for further work.

7.1 Thesis Contributions

The overarching contribution of this research is the development of a model-based assurance framework for SOA. This approach considers the architecture of the SOA system, based on SOA services and the low-level behavioural tasks that are relevant to safety, and features a model-based technique for integrating the design activity and safety analysis. This technique improves traceability and supports safety justification as well as facilitating the maintenance of the safety case and safety analysis artefacts generated within the SOA engineering process. In this thesis, we have focused on four main areas of contribution, namely:

- Identification and adaptation of SOA modelling notations, namely SoaML and BPMN, based on two criteria: the coverage of the structure and behaviour of the services that are deemed suitable for the safety assessment of an SOA design.
- Development of safety analysis techniques for SOA, namely SHA and SFA, which can be used to examine the hazardous failures of services and the failure behaviour of lower-level implementations.
- Development of the concept of modular safety cases for the assurance of safety-critical SOAs, promoting correspondence between the structure of the SOA and the organisation of the modular safety case. This generic argument approach is defined in the form of an SOA Safety Argument

Pattern Catalogue, which can be instantiated to produce safety arguments for specific SOA-based systems.

- Implementation of tool-support that provides automated capabilities for building the SOA models, applying the safety analysis and building the safety case for the system.

Concluding observations regarding these contributions are discussed in the following sections. Finally, in Section 7.2, we suggest some areas for further work.

7.1.1. SOA Modelling

In this thesis, we proposed a systematic, model-based approach to address SOA safety, which is supported by tooling. We call this approach the SOA Safety Assurance Framework. The Framework captures the essential relationships between safety concerns and how they directly relate to SOA services, tasks and flows. The SOA modelling pulls together SOA safety analyses and safety cases in such a way that they are traceably linked during the engineering of safety-critical service-based systems.

7.1.2. SOA Safety Analysis

A key contribution made by the safety analysis techniques developed during our research is to provide structured step-by-step processes for the examination of the design of an SOA. The clear system design models, in SoaML and BPMN, facilitate the systematic integration of safety analysis in the SOA design process, mainly by being fully traceable to the SOA architectural models. In this thesis, we propose and demonstrate safety analysis techniques for examining an SOA design and generating relevant safety requirements which are focussed on the SOA's structure and concerns. These safety requirements provide the basis for integrating safety considerations into the design of the system, by providing a clear rationale and steer for design decisions. This analysis also emphasises the interactions between services and tasks in order to identify failures in participant behaviour that could contribute to service hazards within the SOA system. Our approach provides more transparency in the generation of SOA system safety requirements than do traditional methods, which typically force a distinction between safety and design.

7.1.3. SOA Safety Case Development

This thesis has developed a new approach to developing SOA safety cases through safety case patterns, in order to address the need for explicit safety assurance within SOA engineering. The safety argument patterns presented in Chapter 5 demonstrate how a defensible argument for an SOA system can be constructed by allowing the argument structure to mirror the SOA design, thereby improving the clarity of the reasoning. Specifically, the patterns show how the safety arguments have been structured to correspond directly to the SOA design as defined in the structured architectural models.

7.1.4. Tool-support

The tools implemented in this research are used to support the capabilities of the SOA Safety Assurance Framework. The tools provide a semi-automated environment for supporting the system safety analysis methods and for representing SLAs. They also provide a means for improving traceability between the different SOA models, by linking structural models in SoaML with behavioural models in BPMN.

7.2 Further work

During the course our research, we have identified a number of areas for future work:

- Integration of safety analysis with search-based technologies (e.g. simulation and model-checking);
- Further tool-support to implement model-based assurance cases for SOA based on model weaving;
- Extending the work to cover runtime composition and certification of SOA;
- Extending the SOA safety case pattern catalogue as a generic solution for related domains.

The following sections provide a brief overview of these topics:

7.2.1. Integration of Safety Analysis with Search-based Technologies

One important technique with potential for supporting safety analysis of SOA system, which we have not investigated as part of this thesis, is the search-based approach. One beneficial area of future work would therefore be to investigate different search-based technologies which could potentially provide an automated reasoning basis for SOA safety analysis. This work could include the development of a safety analysis approach based on model-checking, which could on the one hand provide an automated proof that the system satisfies the safety properties (assuming that they could be defined with a suitable formalism) and on the other hand identify scenarios which lead to undesired events. Similarly, simulation models could be built for an SOA during safety analysis, allowing for dynamic exploration of the behaviour of the SOA under different scenarios.

7.2.2. Further tool-support to implement model-based assurance cases for SOA based on model weaving

Various tools have been extended over the course of this research. These tools were developed within the Eclipse Integrated Development Environment platform. We believe that there is scope to add automated capabilities that would improve the efficiency of the argument pattern instantiation process which has been developed based on the work of (164). This approach was not fully evaluated in the case studies, but is a promising basis for future work on model-based approaches to assurance case development. Further work here includes the evaluation of this automated approach via further case studies which would suggest further ways to integrate SOA modelling, safety analysis and safety cases using model-based techniques.

7.2.3. Runtime Composition and Certification of SOA

This thesis has described a process for developing SOA safety cases that demonstrates how a defensible argument for an SOA system can be structured at design-time. One area that was not considered as part of the thesis was the maintenance of the safety analyses and the safety case at runtime, specifically the runtime composition of argument modules to produce a comprehensive safety argument for an SOA. Runtime composition and certification was

explicitly outside of the scope of the research. However, there are key issues in this area which are worthy of further investigation, particularly in terms of their application to autonomous service-based systems. There is some scope for exploring and constructing assurance cases for SOA to support runtime and dynamic composition and certification (165).

7.2.4. Safety Case Pattern Catalogue for SOA as Generic Solution to Related Systems Domains

In this thesis, we have developed an approach to evolving SOA safety cases using the modular and pattern extensions of GSN. We have proposed a safety case pattern catalogue which captures successful safety argument structures which are not specific to any domain. One area for further work is to investigate the applicability of the patterns for the justification of related systems such as Systems of Systems, as discussed in the literature review (Chapter2). Research should be done to investigate how systems composed of different and independent elements can be justified through a generic set of argument patterns.

7.3 Coda

The author hopes that the SOA Safety Assurance Framework created in this thesis, supported by the evaluation and case studies, will deliver valuable and valid insights both for future research and for industrial practice in the use of SOA for safety-critical applications.

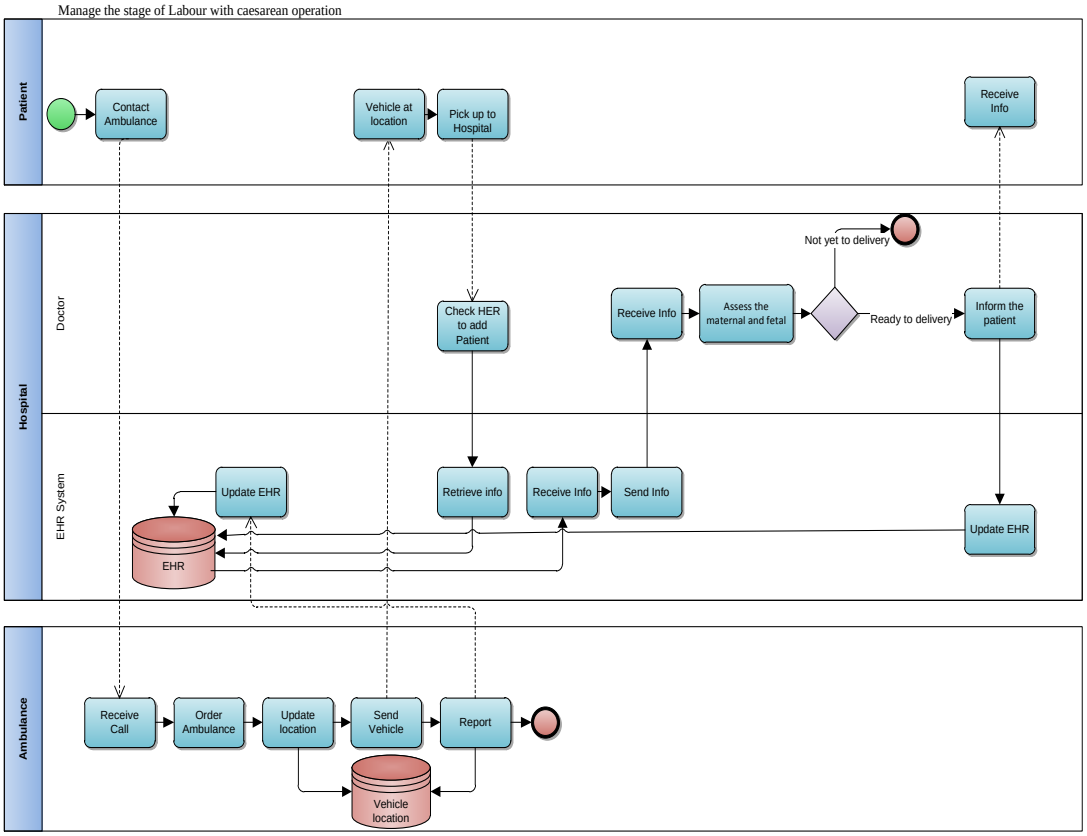
Appendix A: Managing Child Birth and Caesarean

In this medical process, we have applied BPMN to do the modelling for the business processes. The main target of using BPMN is to apply safety analysis and assurance because of BPMN's safety support tool (166).

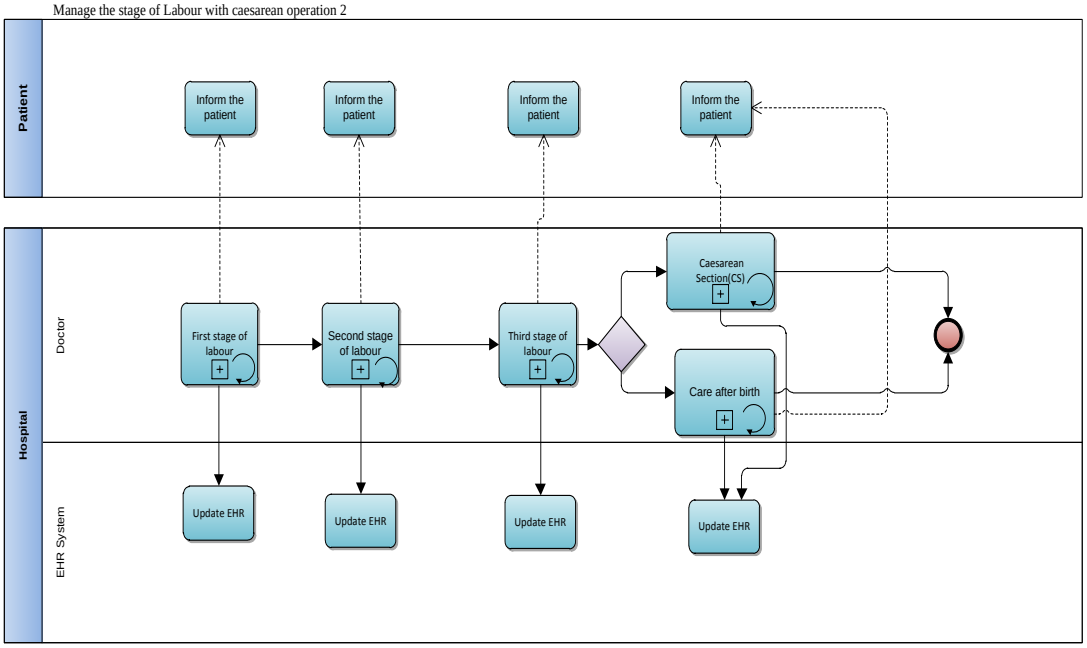
Child birth scenario (137)

- Figure 63, in the beginning the woman contacts the ambulance, which assumes the delivery time is very close.
- The ambulance station orders a vehicle to go and collect the patient and take her to hospital.
- Check if the patient's information is in the EHR system and add her information if she is not in the system.
- Assess the patient, checking the maternal and foetal condition to determine if she is ready to deliver. If not we stop the process, otherwise, we start preparations for labour & care. At each stage, we need to update the EHR system for the patient. As explained in Figure Manage the stage of Labour with caesarean operation.
- We start the first stage as seen in Figure Manage the stage of labour with caesarean operation 2.
- Then, start the examination as shown in Figure First stage of labour
- If there is any delay we apply the Figure of Delay in the First stage and Delay in the First stage (2) Otherwise we move to the Figure second stage of labour.
- In the second stage, we need to apply the Figure Second stage of labour and if there is any delay, we apply the Figure Delay in the Second stage of labour.
- If the second stage, passes successfully, we need to apply the Figure of Third stage of labour and continue the demonstration of the process until we get to figure stage of labour Care after birth, otherwise we will

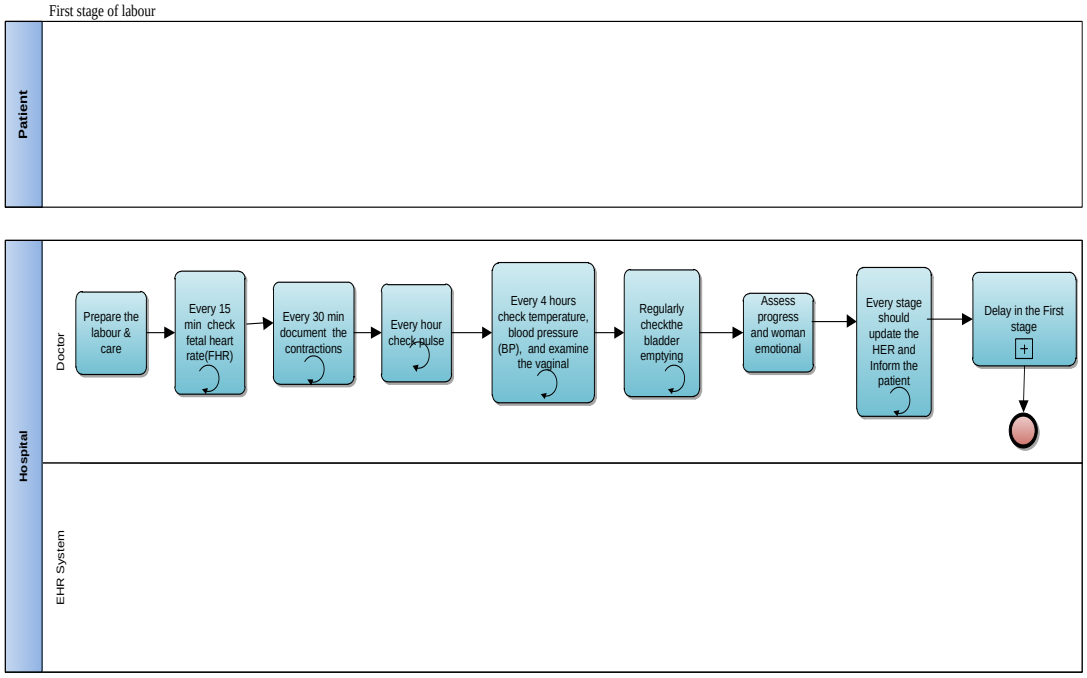
move to Figure stage of consider caesarean section CS and apply the Caesarean Figure.



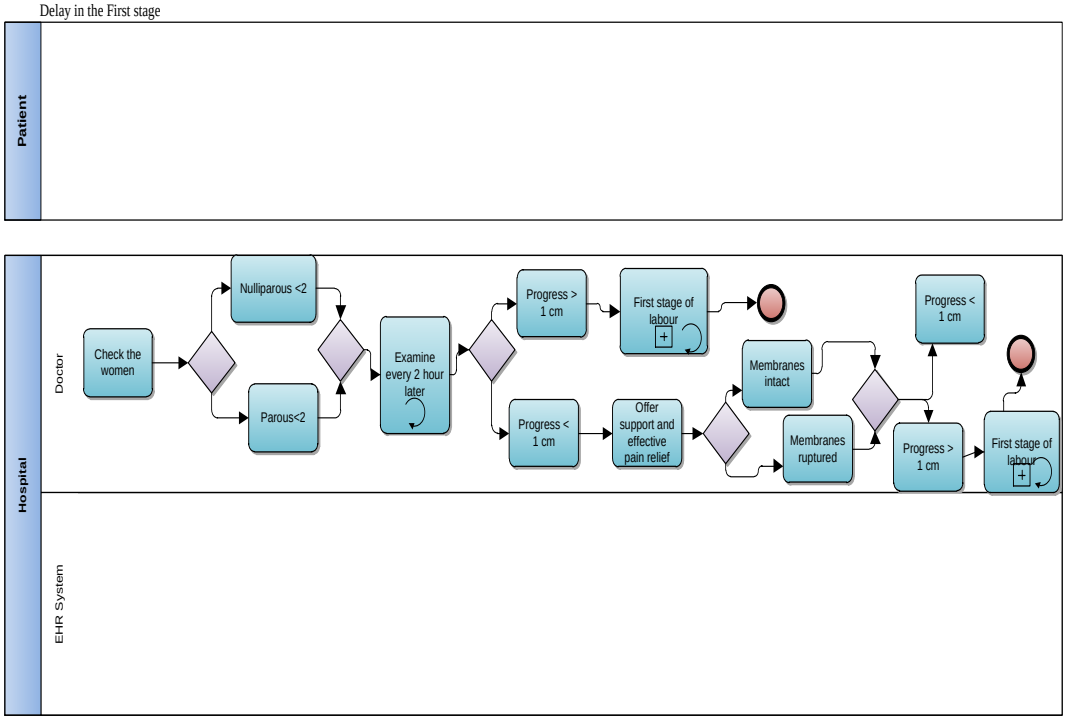
Manage the stage of Labour with caesarean operation 2



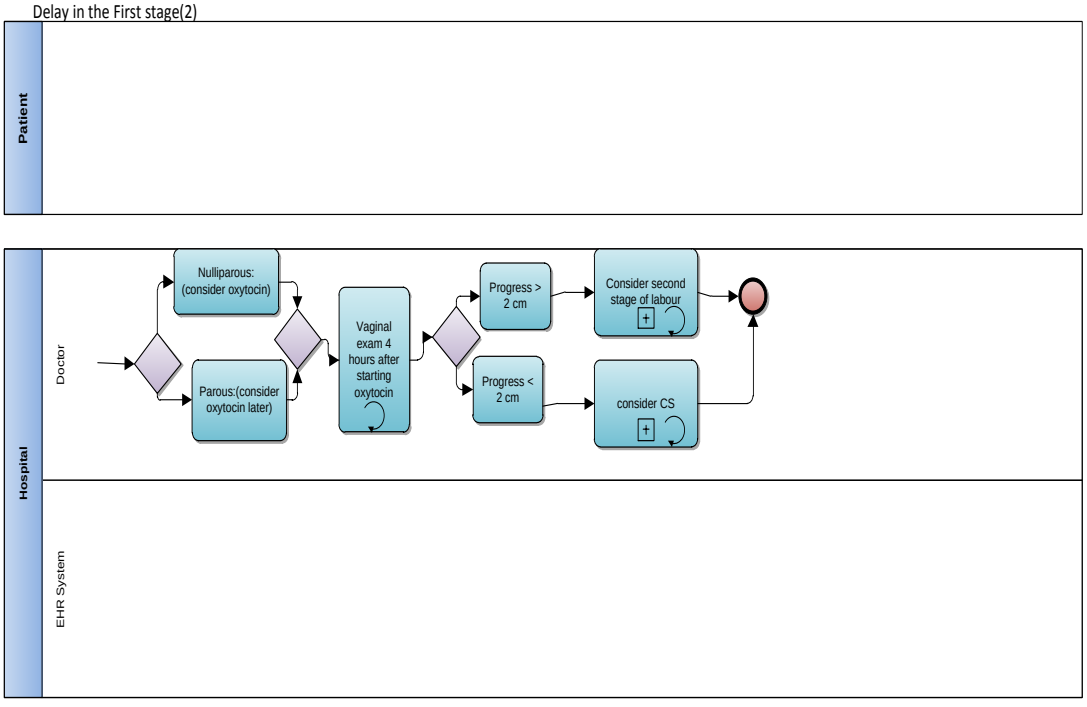
First stage of labour



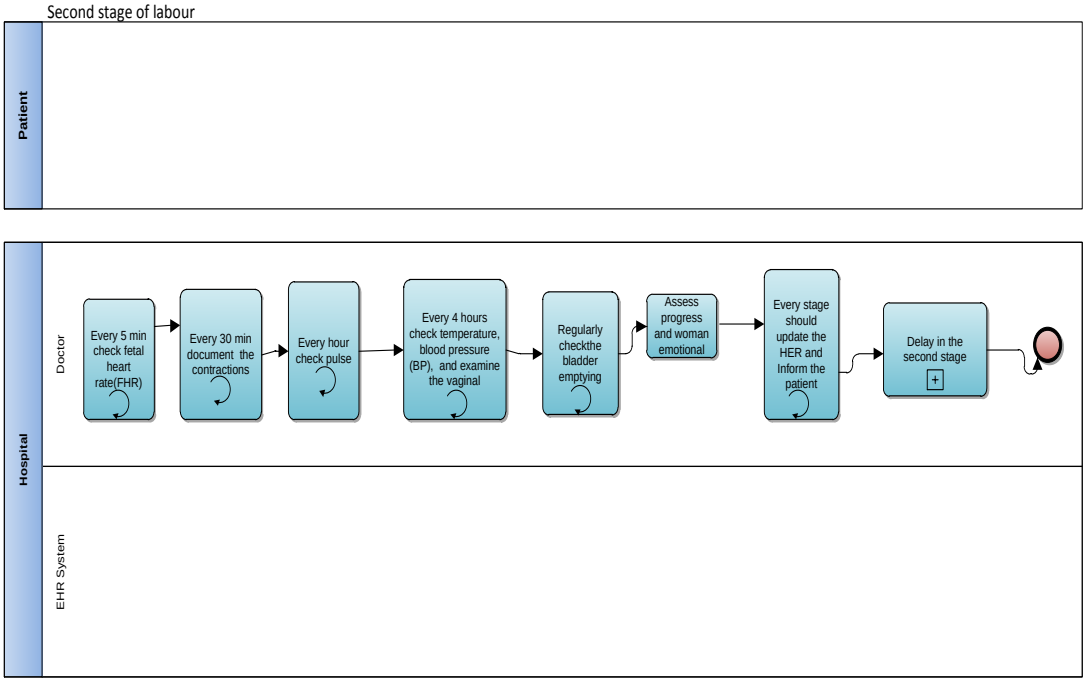
Delay in the First stage



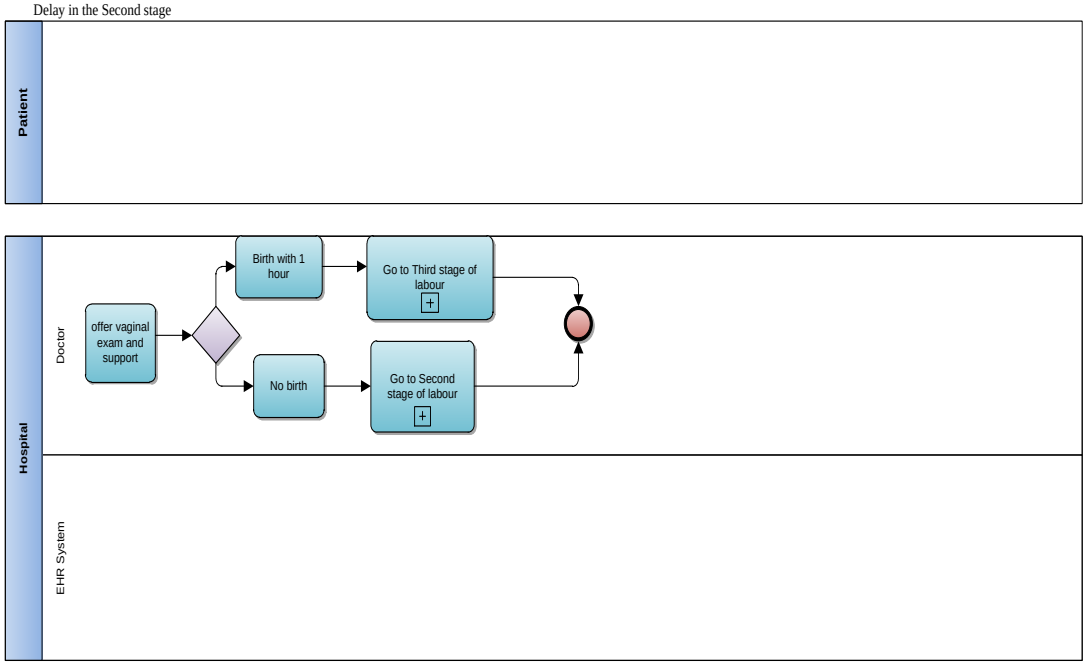
Delay in the First stage(2)

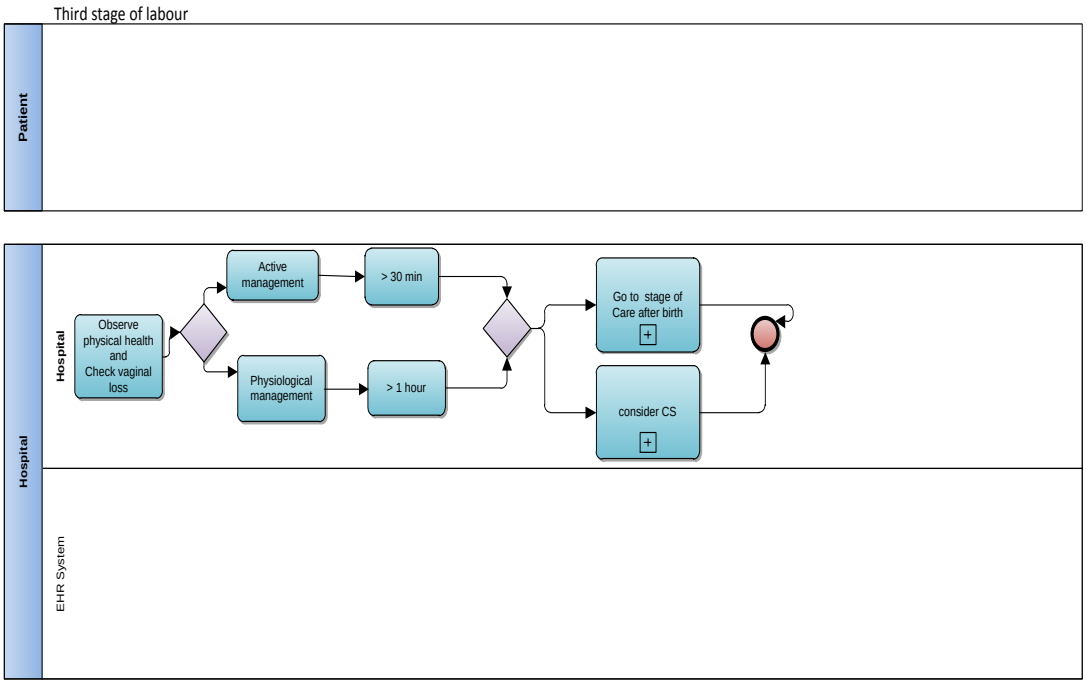


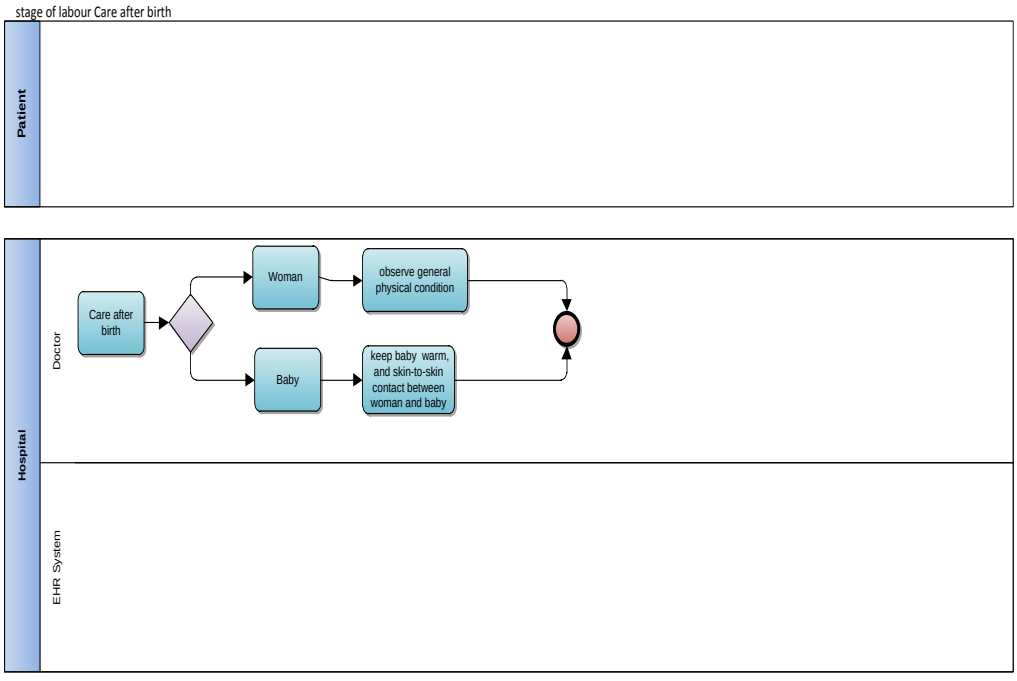
Second stage of labour

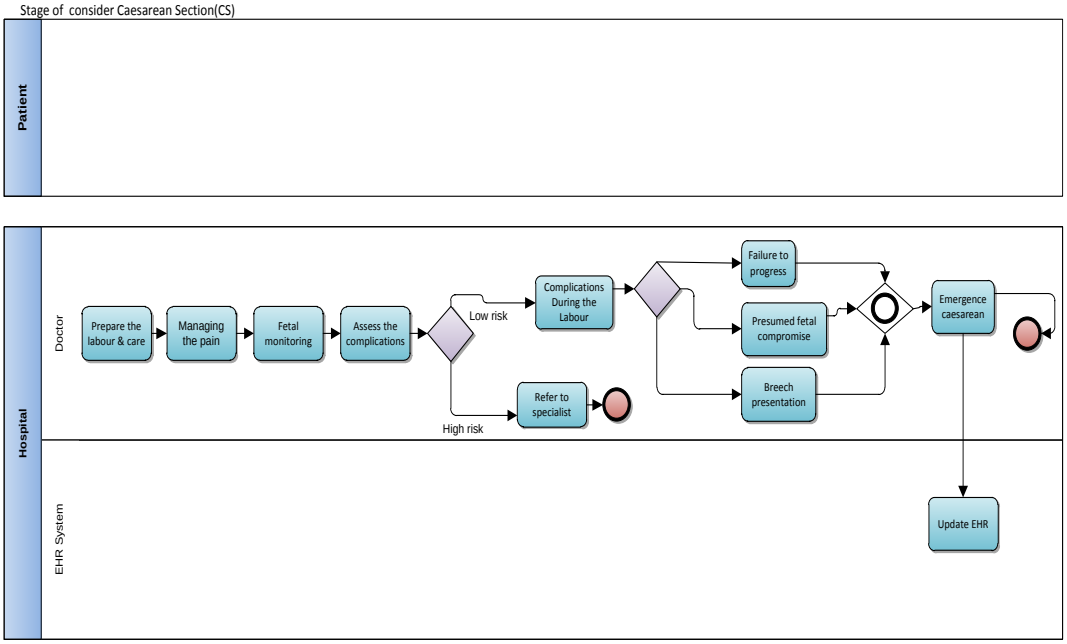


Delay in the Second stage




Third stage of labour
Patient
Hospital
Observe physical health and Check vaginal loss
Active management
> 30 min
Physiological management
> 1 hour
Go to stage of Care after birth
consider CS
Hospital
EHR System


stage of labour Care after birth
Patient
Hospital
Doctor
Care after birth
Woman
observe general physical condition
Baby
keep baby warm, and skin-to-skin contact between woman and baby
EHR System



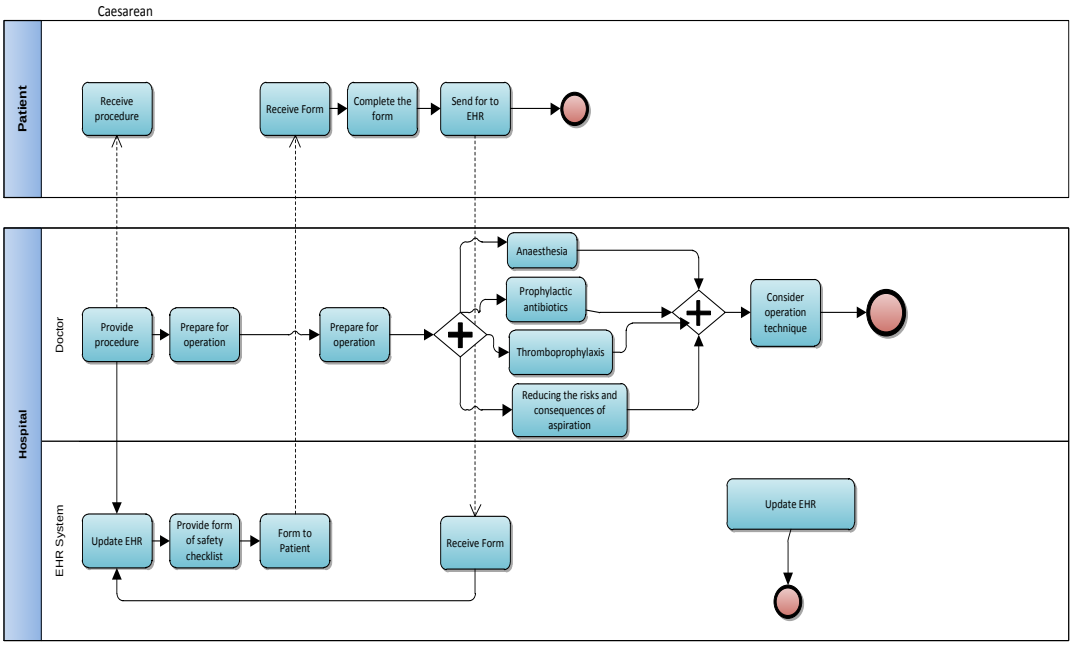


Figure 63: the Medical Process for Child Birth Scenario

Appendix B. SHA for Healthcare Case Study

This section considers the SHA for two services:

(1) Request ambulance;

(2) Update health record;

Service	**Failure Modes**	**Effects**	**Severity**	**SSRs**	**Mitigation**
Request ambulance	Ambulance not requested	Delay in treatment and increased complication	Major	The failure mode "ambulance not request" shall be prevented or mitigated.	Active monitoring and cross-checking between calling and requesting ambulances to ensure the communication is applied successfully
	Ambulance provided when not required	N/A No direct safety effects but waste for critical services.	N/A	None.	None.
	Ambulance required later than intended	Delay in treatment, and increased complication	Major	The failure mode "ambulance required later than intended" shall be prevented or mitigated.	Active monitoring of timing targets and strategies for recovery from timing-related failures.
	Ambulance request improper information	Delay in treatment, and increased complication	Major	The failure mode "Ambulance request improper information" shall be prevented or mitigated.	Early checking information and confirmation between calling and requested ambulances to ensure the information transfer correctly through the system

Table 38: A sample of SHA output for the request ambulance service system

Service	Failure Modes	Effects	Severity	SSRs	Mitigation
Update health record	Loss of Updating	Difficulty in assessing condition	Considerable	The failure mode "Loss of Updating" shall be prevented or mitigated.	Use of redundancy through fitting an addition flow to provide patient information from the another source of health record
	Updating provided when not required	Incorrect data leading to wrong treatment	Major	The failure mode "Updating provided when not required" shall be prevented or mitigated.	Active cross-checking and interlocks
	Updating provide later than intended	Delay the treatment, and increased complication	Considerable	The failure mode "Updating provide later than intended" shall be prevented or mitigated.	Active monitoring of timing targets
	Inaccurate service Updating	Incorrect treatments	Major	The failure mode "Inaccurate service Updating" shall be prevented or mitigated.	Active monitoring and cross-checking to ensure the record updated correctly

Table 39: A sample of SHA output for the Update health record service system

Appendix C. SFA for Healthcare Case Study

This section considers the SFA for other flows:

(1) From call ambulance and provide information to receive request;
(2) From authorize and authenticate to provide patient name;

ID (Flow)	Guide Word	Deviation	Possible Causes	Effects	DSSRs	Mitigation
1.1	Omission (Incomplete data)	no request received	Deployment fault (Request resource missing) Service/server incompatible Task crashed	Incomplete data, no information for the request	The failure mode "no output process is sent" shall be prevented or mitigated.	Ensure redundancy in the system can help to duplicate the task within the system which allows one or more copies of the same task
1.2	Commission (Repetition-Data overrun)	With additional output	incorrect or mismatching messages description	No safety effect Unwanted change	N/A	N/A
1.3	Early	N/A	N/A	N/A		N/A
1.4	Late	Unacceptable response time	Failure in scheduling system Time out of the task	Overloading in task in the flow may lead to delay tasks	The failure mode "Unacceptable response time to change in demands" shall be prevented or mitigated.	Monitoring for timing failure and developing recovery strategies
1.5	Value	Incorrect request	Incorrect input or conversion fault execution fault, incorrect result or servicer crash	Delay in the treatment and increase complication	The failure mode "One component of output is not proportional to demand" shall be prevented or mitigated.	Diversity to make sure there is not error propagation across system boundaries.

Table 40: A sample of SFA output for flow from call ambulance and provide information to receive request

ID (Flow)	Guide Word	Deviation	Possible Causes	Effects	DSSRs	Mitigation
1.1	Omission	No access available	authorization and authentication failed	No patient history and background	The failure mode "No access available" shall be prevented or mitigated.	Use of redundancy in data sources for access the system
1.2	Commission	unwanted access	Failure of output process on the system.	No direct safety effects	The failure mode "unwanted access" shall be prevented or mitigated.	Monitoring of access requests
1.3	Early	N/A	N/A	N/A		Just: the output cannot be produced
1.4	Late	Unacceptable response time	Failure in scheduling	Overloading	The failure mode "Unacceptable response time" shall be prevented or mitigated.	Monitoring of time delays and recovery by giving more priority for delayed requests
1.5	Incorrect access the system	Incorrect output or task crash	Undetected failure in any process. insufficient security	unknown allergy status	The failure mode "Incorrect output or task crash " shall be prevented or mitigated.	Data entry cross-checking, online monitoring and fault containment

Table 41: A sample of SFA output for flow from provide user name and password to authorize and authenticate

Appendix D. Safety Case for Healthcare Case Study:

This section considers the safety arguments for another specific service: Retrieve Health Record. The safety arguments are described in this Appendix.

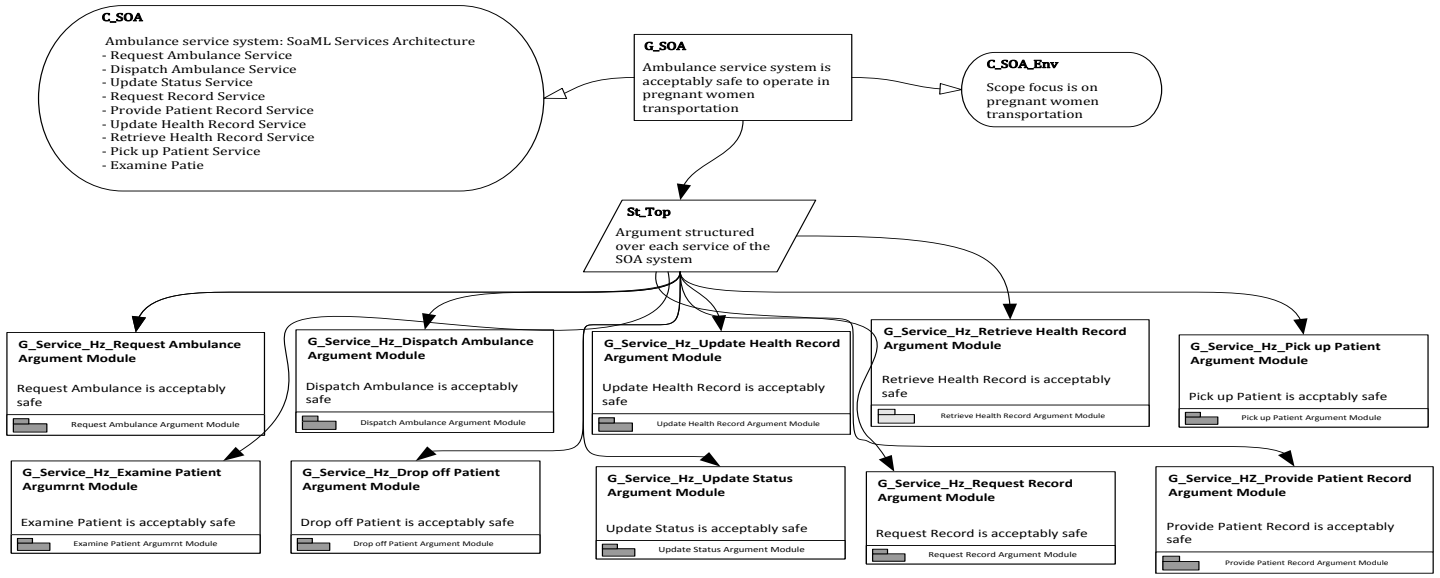


Figure 64: Top Argument Module

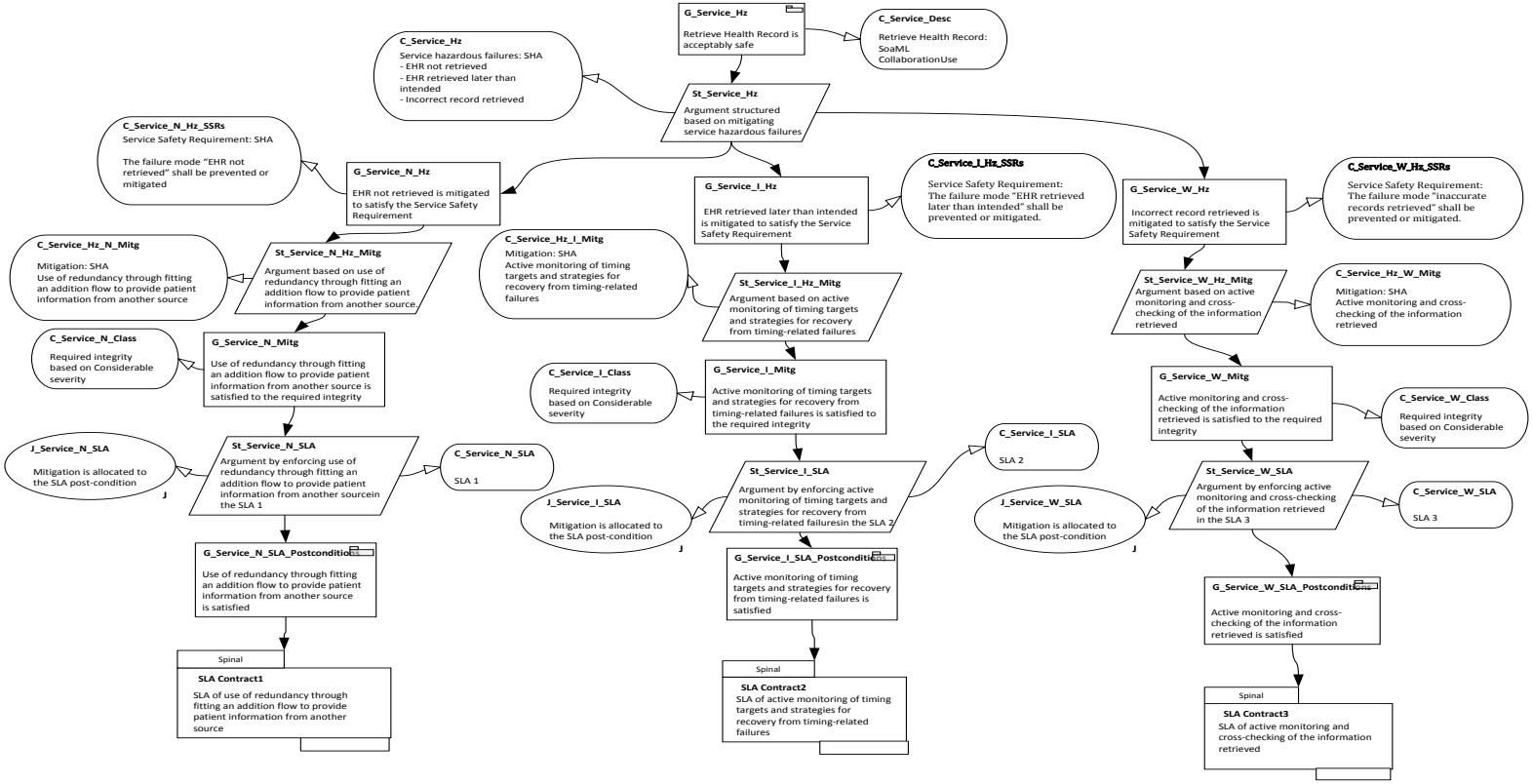


Figure 65: Service Argument for Retrieve Health Record

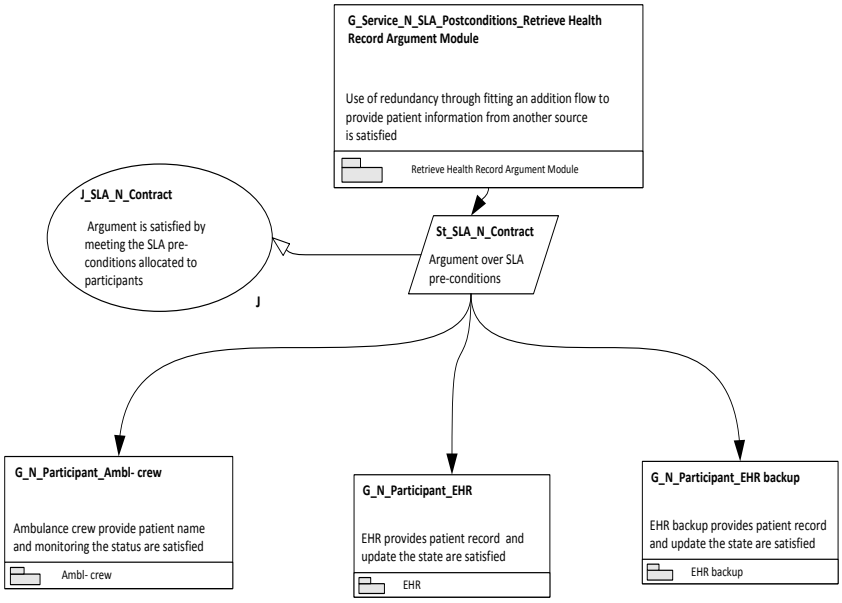


Figure 66: SLA contact Argument for Retrieve Health Record

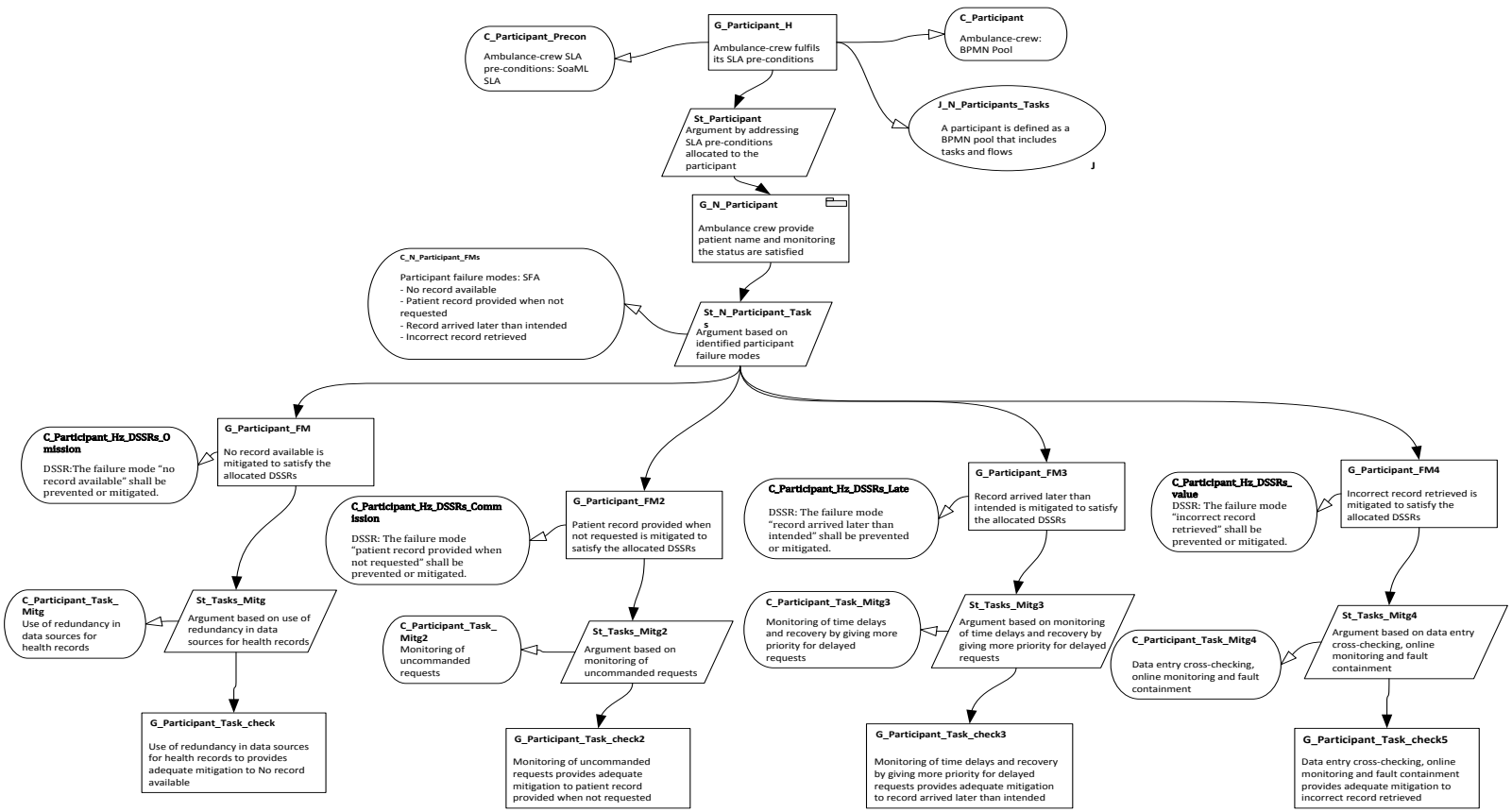


Figure 67: Participant Argument for Retrieve Health Record

Appendix E. Natural Gas Processing Case Study

SFA

(1) Measure sweet gas by analyser to confirm water content is less than %1; and

(2) Pump the liquid gas to receive liquid gas.

ID (Flow)	Guide Word	Deviation	Possible Causes	Effects	Justification/Design Recommendations
1.1	Omission (no communication within the flow)	No output data is sent	Description incomplete or incorrect input Service crashed	Incomplete data	Use of redundancy in data sources
	Omission (Incomplete flow)	Part of output signal is missing	Required resource missing	Inconsistent water content	Checking integrity of data and monitoring
1.2	Commission (Repetition)	With additional output process sent. No change is required.	False flow Execution fault (Incorrect result)	Conflicting request received Unwanted change	Use of interlock mechanism
	Commission (Spurious)	The communication is unintended	Conversation fault (Incorrect input) (server crashed)	Conflicting request received Unwanted change	Use of interlock mechanism
1.3	Early	N/A	N/A	N/A	N/A
1.4	Late	Unacceptable response time to changes in demands	Execution fault Time out (severe crash or communication failure)	Late detection of safety events	Active monitoring of timing targets and strategies for recovery from timing-related failures through analyser

1.5	Value	One component of output is not proportional to demand	Require resource missing Range exceeded (execution fault incorrect result- or servicer crash)	False value Unknown value	Applying diversity in the flow and early cross-checking fault containment to detect and reject incorrect value and fault containment.

Table 42: Measure sweet gas by analyser to confirm water content is less than %1;

ID	Guide Word	Deviation	Possible Causes	Effects	Justification/Design Recommendations
1.1	Omission (no communication within the flow)	no output process is sent	Content fault (conflicting Content)	Liquid not received	Use of redundancy through providing other sources
	Omission (Incomplete flow)	part of output signal is missing	Required resource missing	Inconsistent receiving of liquid gas	Checking and monitoring and fault containment
1.2	Commission (Repetition)	With additional output process sent. No change is required	Content fault (Redundant description)	Overloading	Check high-integrity among participants to consider sufficiency pump the liquid
	Commission (Spurious)	The communication is unintended	conversation fault (Incorrect input)	Overloading	Apply high-integrity among participants to consider sufficiently incorrect pump liquid
1.3	Early	N/A	N/A	N/A	N/A
1.4	Late	Unacceptable response time to change in demands	Execution fault Time out communication failure)	Delays in production	Active monitoring transmitter of timing targets and strategies for recovery from timing-related failures
1.5	Value	One component of output is not proportional to demand	Execution fault	Incorrect level of liquid gas	Increased monitoring + fault containment

Table 43: Pump the liquid gas to receive liquid gas

Example of Safety Case for Natural Gas Processing for the Absorb Water Service

Absorb water (Top)

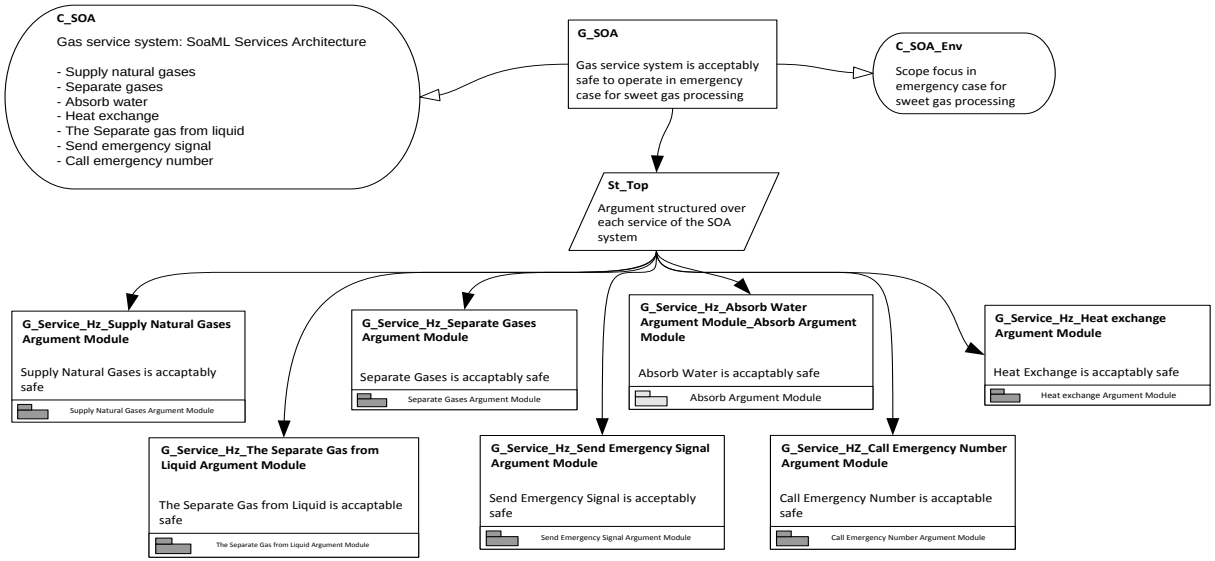


Figure 68: Top Argument Module for Absorb water

Absorb water Service

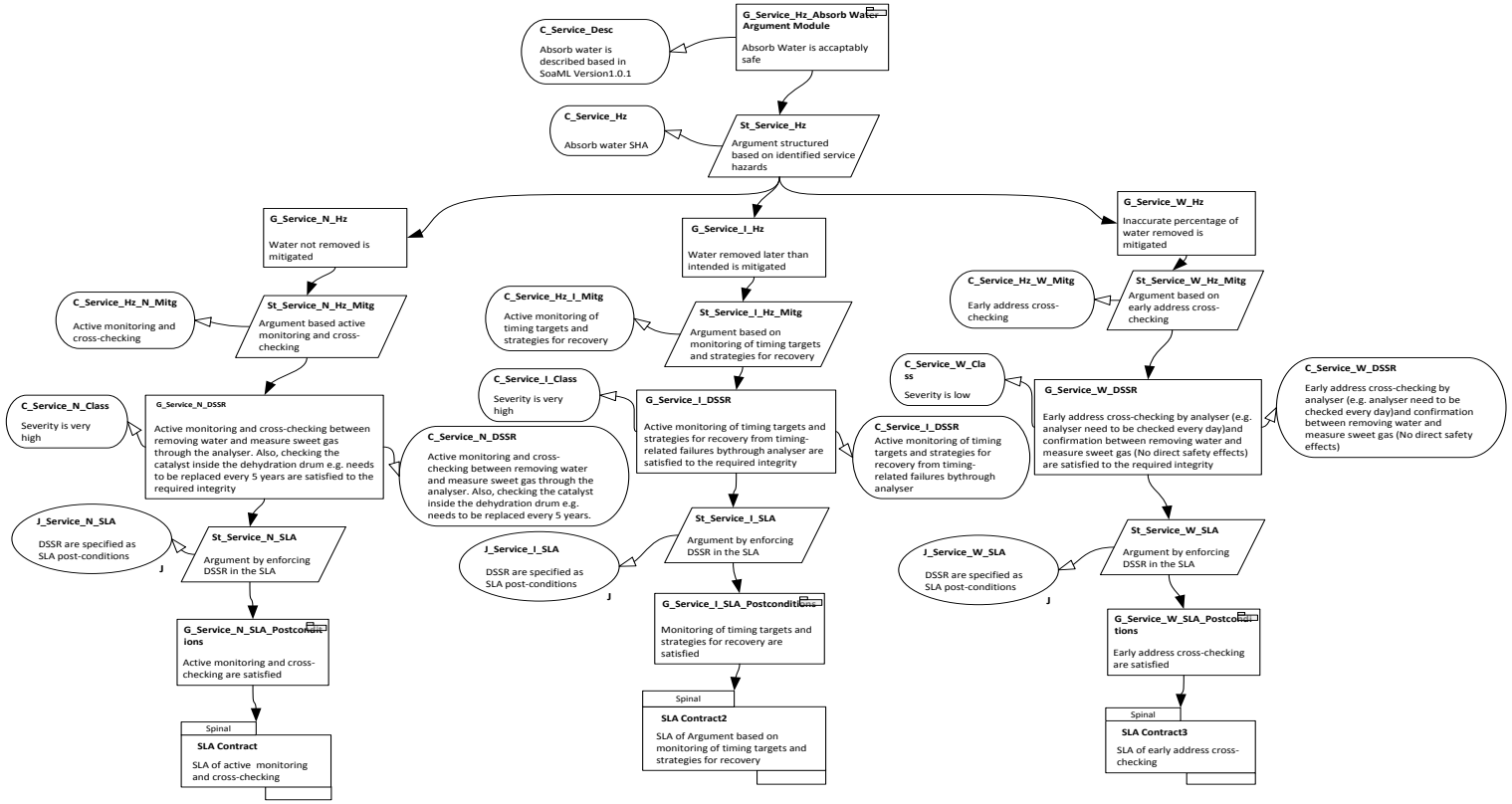


Figure 69: Service Argument for Absorb Water Service

SLA Contract

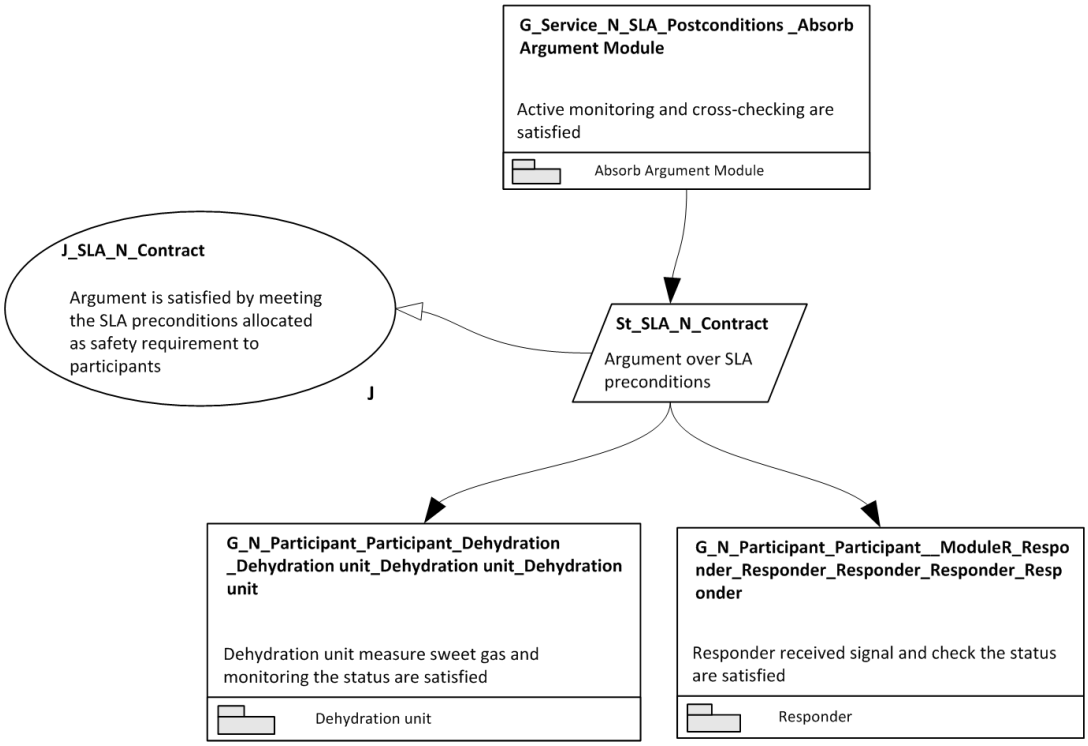


Figure 70: SLA contact Argument for Absorb Water Service

Participant

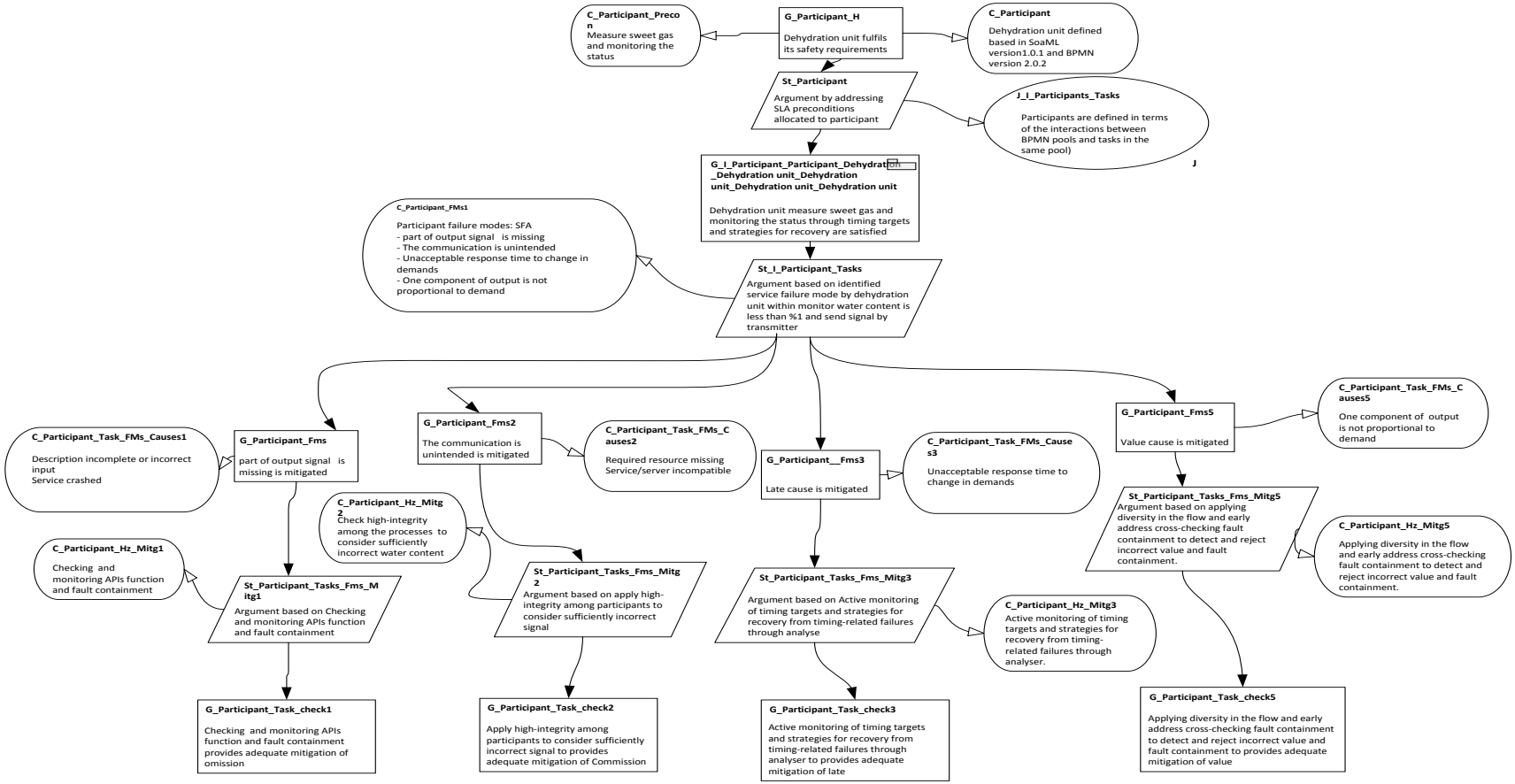


Figure 71: Participant Argument for Absorb Water Service

Appendix F. Interviews

INFORMATION SHEET

PhD PROJECT TITLE: Service-Oriented Architectures for Safety Critical Systems

INVITATION

You are being asked to take part in a research study for a PhD project on the safety assurance of service-oriented architectures.

WHAT WILL HAPPEN

In this study, you will be asked a set of predefined questions and your answers will be recorded on paper at the time and written up on electronic media after the interview.

TIME COMMITMENT

The study typically takes 20 minutes.

PARTICIPANTS' RIGHTS

You may decide to stop being a part of the research study at any time without giving an explanation. You have the right to ask that any data you have supplied to that point be withdrawn/destroyed.

You have the right to omit or refuse to answer or respond to any question that is asked of you.

You have the right to have your questions about the procedures answered (unless answering these questions would interfere with the study's outcome). If you have any questions as a result of reading this information sheet, you should ask the researcher before the study begins.

BENEFITS AND RISKS

There are no known benefits for me or risks for you in this study.

COST, REIMBURSEMENT AND COMPENSATION

Your participation in this study is voluntary.

CONFIDENTIALITY/ANONYMITY

The data I collect will not contain any personal information about you except for you current role and your present and previous experience.

The results of the interview will be written up as part of my doctoral thesis. They may also be published in academic outlets such as journals, conferences or academic books. In all cases, the data will only be presented in summary form and you will not be directly identifiable in any way.

FOR FURTHER INFORMATION

Abdulaziz Al-Humam will be glad to answer your questions about this study at any time. He may be contacted as follows

Abdulaziz Al-Humam, aaah501@york.ac.uk

If you want to find out about the final results of this study, please let me know now and I will email you the results upon completion.

CONSENT FORM

PhD PROJECT TITLE: Service-Oriented Architectures for Safety Critical Systems

Name of Researcher: Abdulaziz Al-Humam

Supervisors Ibrahim Habli

Please initial box

1. I confirm that I have read and understand the information sheet for the above study. I have had the opportunity to consider the information, ask questions and have had these answered satisfactorily. ☐

2. I understand that my participation is voluntary and that I am free to withdraw at any time, without giving any reason. ☐

3. I understand that any information given by me may be used in future reports, articles or presentations by the researcher. ☐

4. I understand that my name will not appear in any reports, articles or presentations. ☐

5. I agree to take part in the above study. ☐

_____	_____	_____
Name of Participant	Date	Signature
_____	_____	_____
Researcher	Date	Signature

Date: Areas of expertise of the interviewee:
7- What do you think about the overall approach?
8- Do you think that the modelling approach improve the understanding of the system?
a. What aspects in particular did you find useful?
b. What aspects in particular did you find hard to use?
9- Do you think that the safety analysis methods, SHA and SFA, can provide a systematic approach to analysing services?
a. What aspects do you think have the potential to improve practice?
b. What aspects do you think are infeasible to use?
10- Do you think that the safety case used improve the justification for the safety of the service-based system?
a. What aspects do you think have the potential to improve practice?
b. What aspects do you think are infeasible to use?
11- Do you think that the modelling, safety analysis and safety case artefacts are clearly integrated?
12- Can you provide suggestions for areas that can improve the framework?

Table 44: Interview Questions

The presentations of the case studies respectively (Healthcare - Natural Gas Processing)

Service-Oriented Architectures for Safety Critical Systems

HISE Research Group
Abdulaziz Al-Humam
Supervisor: Ibrahim Habli

June 2015

1

Outline

- Motivation
- Research Hypothesis and Contributions
  - SOA Modelling
  - SOA Safety Analysis
  - SOA Safety Cases
- Evaluation
- Conclusions

2

Background

- Most organisations are based on providing and consuming services
- Most well-known service-based sectors include:
  - Healthcare
  - Banking
  - Retail
  - Transport
  - Emergency
- Services in high-risk sectors can make a direct contribution to safety
  - i.e. failure of services can lead to harm

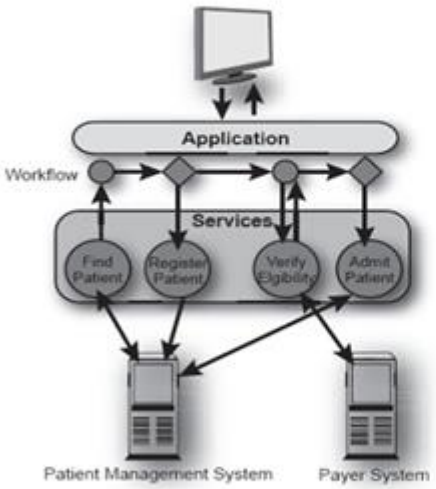
3

Service-oriented Architectures (SOA)

- A **service** is a value delivered to another through a well-defined interface and available to a community
- **SOA** is as a paradigm for defining how people, organisations, and systems provide and use services to achieve results
  - Tasks and data (information flows) are exposed in the form of services
  - Services are used by many applications

4

Example SOA



A conceptual view of healthcare services

Girish Juneja, Blake Dournaee, Joe Natoli & Steve Birkel, 2009

5

Motivation

- To integrate safety analysis and assurance into the engineering of systems designed based on SOA
  - Shortage of research on use of SOAs for development of safety-critical systems
  - Lack of systematic safety analysis processes and methods that address the special features of SOAs
  - Lack of safety assurance strategies for justifying the acceptably safety of SOA

6

Research Hypothesis

Integrated architectural modelling and analysis of safety-critical service-oriented systems provides a **traceable**, **consistent** and **systematic** means for **assuring** the safety of these systems.

7

Research Hypothesis

Integrated architectural modelling and analysis of safety-critical service-oriented systems provides a **traceable**, **consistent** and **systematic** means for **assuring** the safety of these systems.

- **Integrated:** safety analysis and design decisions evolve together
- **Traceable:** forward and backward model-based traceability between design and safety analysis
- **Consistent:** design features are reflected in design and vice versa
- **Systematic:** methodical steps for performing safety assurance
- **Assuring:** explicit reasoning about the safety properties and behaviour

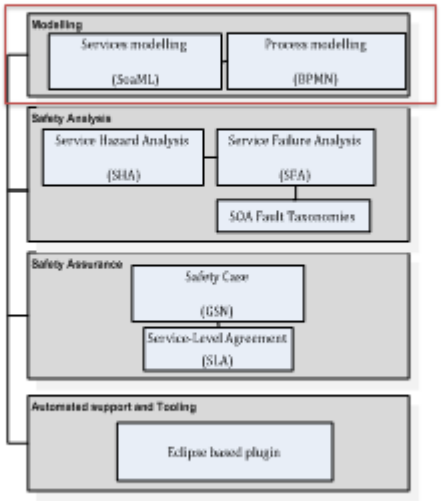
8

Research Contributions

- **Safety Analysis for SOA**
  - Method for analysing the structural view of SOA
    - Service Hazard Analysis (SHA)
  - Method for analysing the behavioural view of SOA
    - Service Failure Analysis (SFA)
- **Safety Case Development for SOA**
  - Modular safety argument patterns for reasoning about SOA safety
- **Automated tool-support**
  - Implementing model-based integration

9

Approach Overview



10

SOA Modelling

Structure and behaviour of SOA represented in two modelling languages:

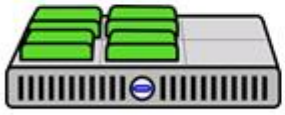
- Service oriented architecture Modeling Language **(SoaML)**
  - Representation of high-level structure of SOA design
  - Provides ability to specify and analyse requirements for each service
- Business Process Modelling Notation **(BPMN)**
  - Captures the dynamic behaviour implemented in the SOA design
  - Provides the capability to communicate and represent the interaction between detailed service tasks

11

Healthcare Case Study

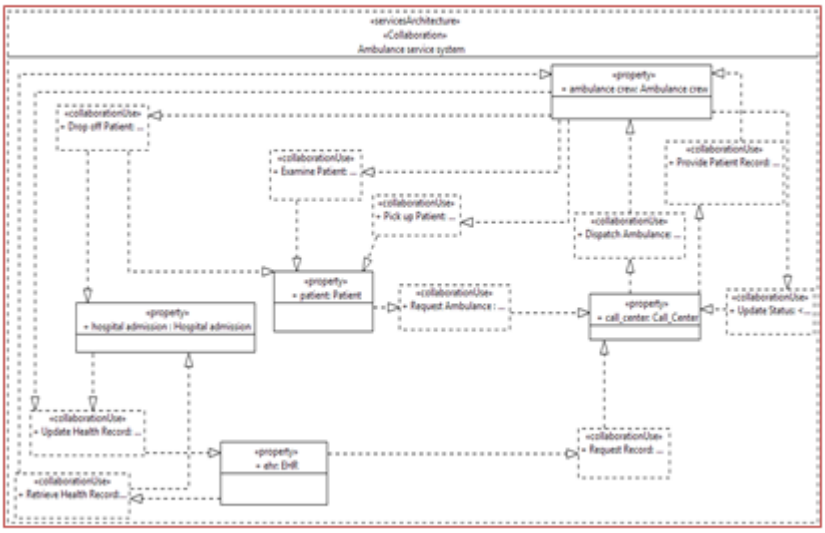
Exploratory case study based on three services:

- Ambulance
- Electronic health records
- Childbirth services



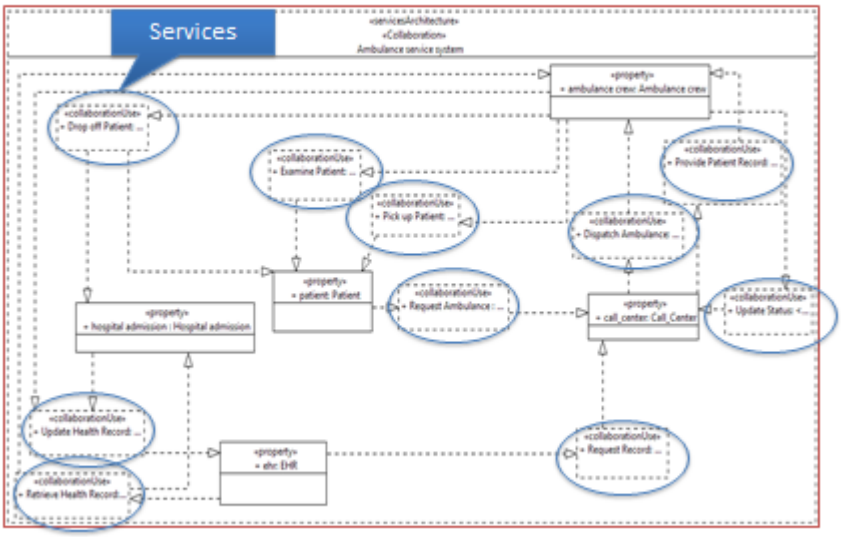
12

Healthcare Case Study: SoaML Services Model



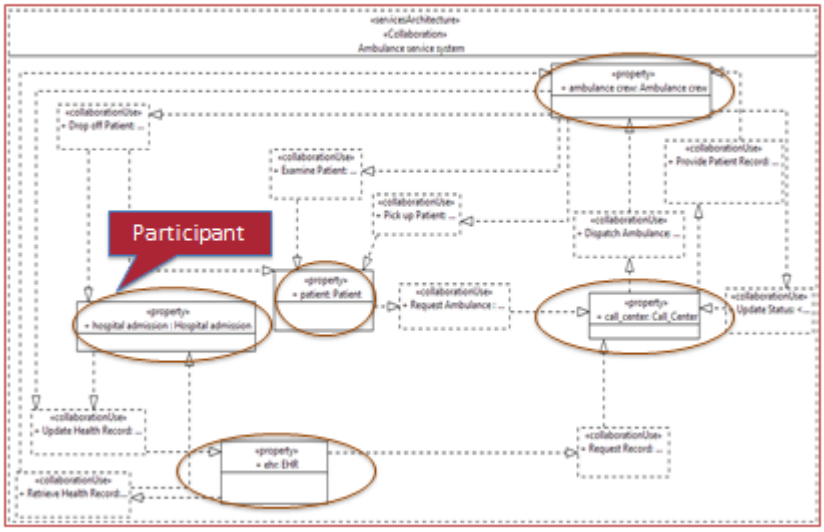
13

Healthcare Case Study: SoaML Services Model



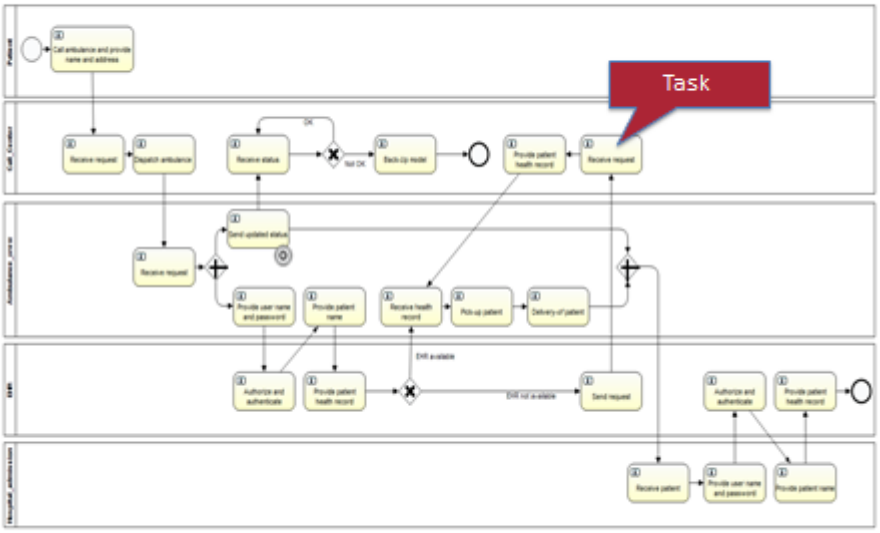
14

Healthcare Case Study : SoaML Services Model



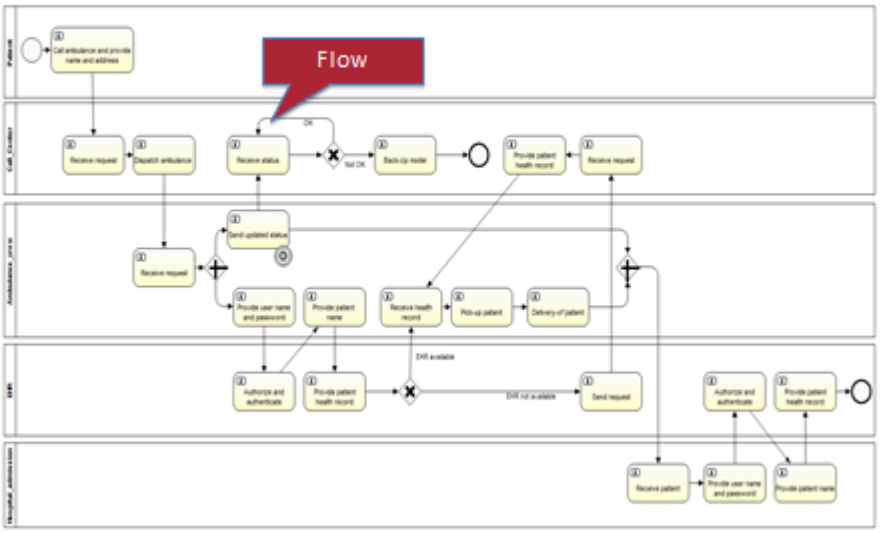
15

Healthcare Case Study: BPMN Model



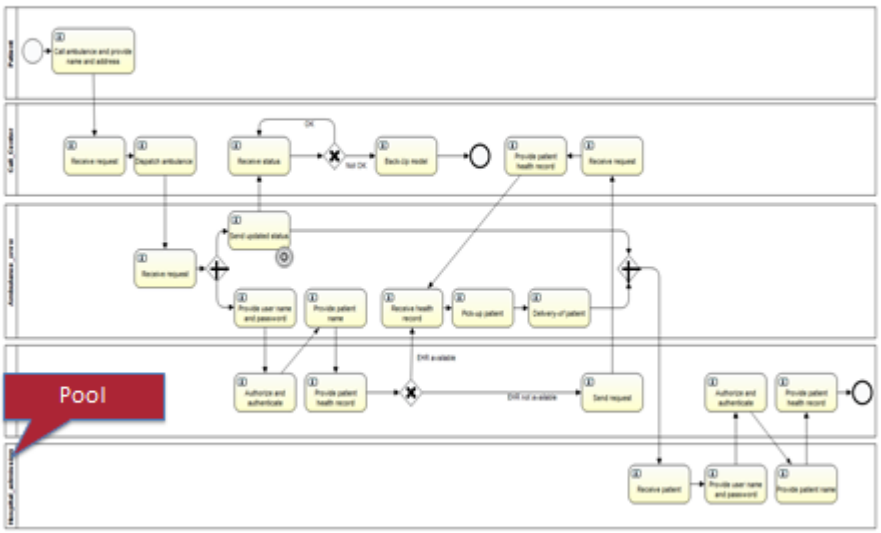
16

Healthcare Case Study: BPMN Model



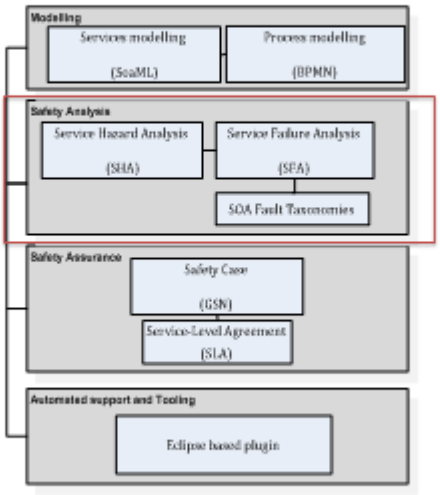
17

Healthcare Case Study: BPMN Model



18

Approach Overview



19

SOA Safety Analysis

- Method for analysing the structural view of SOA
  - Service Hazard Analysis (SHA)
- Method for analysing the behavioural view of SOA
  - Service Failure Analysis (SFA)
  - SOA Fault Taxonomies in SFA
    - Causes of each of the deviations is linked to a specific type of fault defined in SOA fault taxonomy

20

Service Hazard Analysis (SHA)

- A variant of Functional Hazard Assessment (FHA), called Service Hazard Analysis (SHA)

- Analysis based on three potential service deviations
  - (1) Service not provided when required
  - (2) Service provided when not required
  - (3) Incorrect service

Output of SHA is a set of safety requirements defined at the service level

21

SHA Steps

1. Identify a service
2. Identify the service failure modes
3. Determine the safety effects of each service failure mode
4. Determine the safety severity/classification of each service failure mode
5. Provide service safety requirements
6. Identify potential mitigation measures for meeting the service safety requirements

Case Study: SHA Results

Service	Failure Modes	Effects	Severity	SSRs	Mitigation
Dispatch Ambulance Context: *birth before attendance and the patient is actively bleeding following birth*	Ambulance not dispatched	Death	Major	The failure mode "ambulance not dispatched" shall be prevented or mitigated.	Active monitoring and cross-checking between requested and dispatched ambulances.
	Ambulance dispatched when not required	N/A No direct safety effects but waste for critical services.	N/A	None.	None.
	Ambulance dispatched later than intended	Severe morbidity (hypovolaemia, renal failure, cardiac arrest, disseminated intravascular coagulopathy…)	Major	The failure mode "ambulance dispatched later than intended" shall be prevented or mitigated.	Active monitoring of timing targets and strategies for recovery from timing-related failures.
	Ambulance dispatched to the wrong address	Severe morbidity (hypovolaemia, renal failure, cardiac arrest, disseminated intravascular coagulopathy…)	Major	The failure mode "ambulance dispatched later than intended" shall be prevented or mitigated.	Early address cross-checking and confirmation between requested and dispatched ambulances

23

Service Failure Analysis (SFA)

- Analysing the flow of information between interacting tasks
- SHARD safety analysis technique is adapted to analyse the flow of information between the tasks represented
- The output of this analysis is a set of Derived Service Safety Requirements (DSSRs)

24

SFA Steps

1. Identify a flow between two tasks
2. Identify flow failure modes
3. Determine the potential causes of each flow failure mode
4. Determine the potential effects of each flow failure mode
5. Define DSSRs
6. Identify potential mitigation measures for meeting the DSSRs

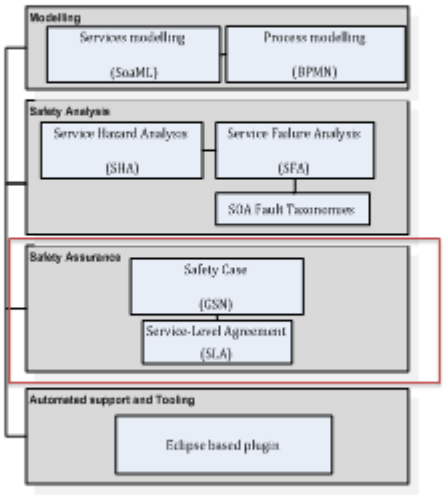
25

Case Study: SFA Results

ID (Flow)	Guide Word	Deviation	Possible Causes	Effects	DSSRs	Mitigation
Flow between 'Dispatch ambulance and 'Receive request'	Omission	No request for ambulance received	Deployment fault (Request resource missing) Service/server incompatible Task crashed	Ambulance not dispatched	The failure mode "no request for ambulance received" shall be prevented or mitigated.	Use of redundancy in data sources by Fit addition flow to the request
	Commission	Uncommanded request for ambulance received	False request, incorrect request Execution fault (Incorrect result)	No safety effect	N/A	N/A
	Early	N/A	N/A	N/A	N/A	N/A
	Late	Request arrived later than intended	Failure in scheduling/ synchronisation Time out related to clock time with effects on other tasks	Ambulance dispatched later than intended	The failure mode "request arrived later than intended" shall be prevented or mitigated.	Monitoring for timing failure and developing recovery strategies
	Value	Incorrect request	Incorrect input or conversion fault execution fault, incorrect result or servicer crash	Delay in the treatment	The failure mode "incorrect request" shall be prevented or mitigated.	Apply diversity via programming solutions to detect and reject improper values and fault containment

26

Approach Overview



27

SOA Safety Cases

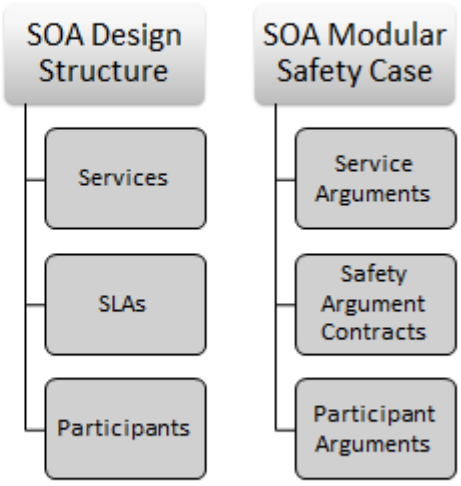
- Assuring services and their design in SOA safety cases
  - Safety argument is structured based on justification of individual services and their interactions through participants
    - Interaction influenced by pre-defined Service-Level Agreements (SLAs)

- Development of modular safety case patterns for SOA
  - SOA argument patterns catalogue includes rules for pattern composition and traceability to the architectural design (i.e. SoaML and BPMN models) and analysis (i.e. SHA and SFA)

28

SOA and Modular Safety Cases

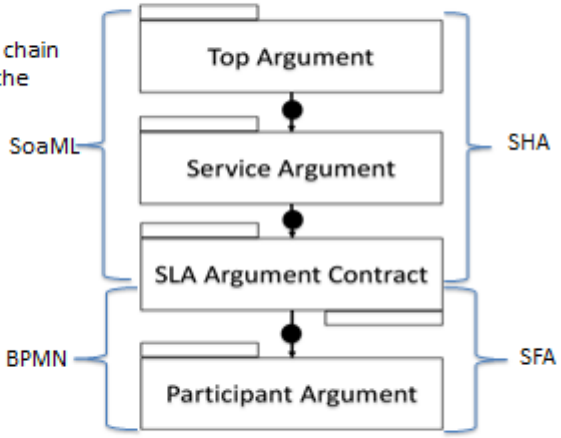
- Correspondence between the high-level SOA design and the overall SOA safety case
- The mapping between SLAs and the argument contracts drives the overall structure of the SOA safety argument and offers traceability between the design and the safety assurance



29

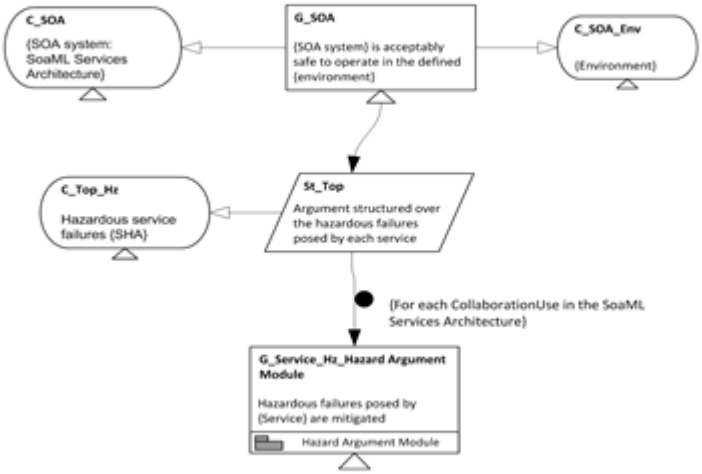
SOA Argument Patterns Catalogue

SOA safety argument pattern catalogue follows a reasoning chain from the Top Argument until the Participant Argument



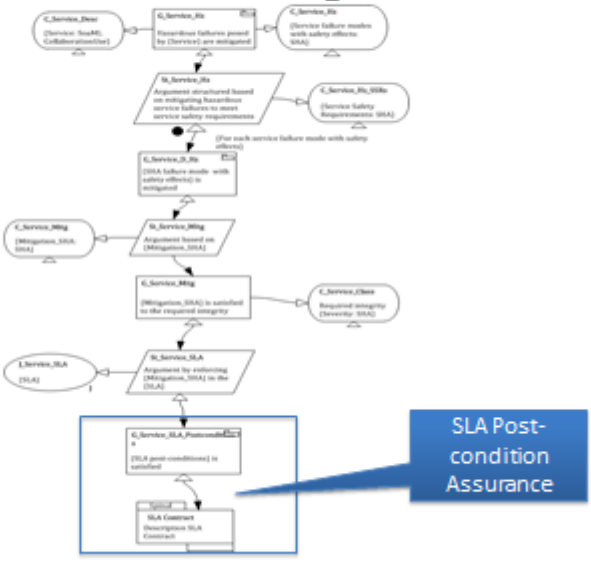
30

Top Argument



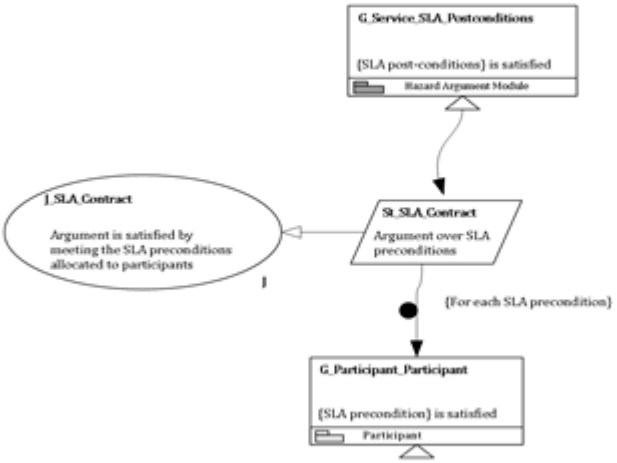
31

Service Hazard Argument



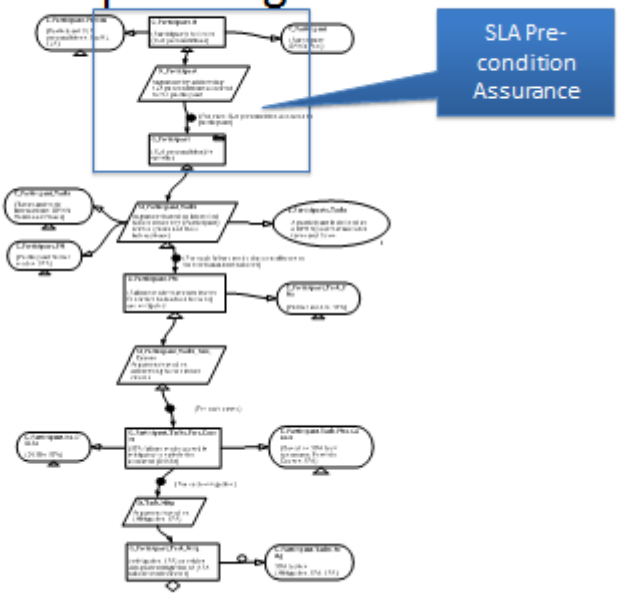
32

SLA Contract Argument



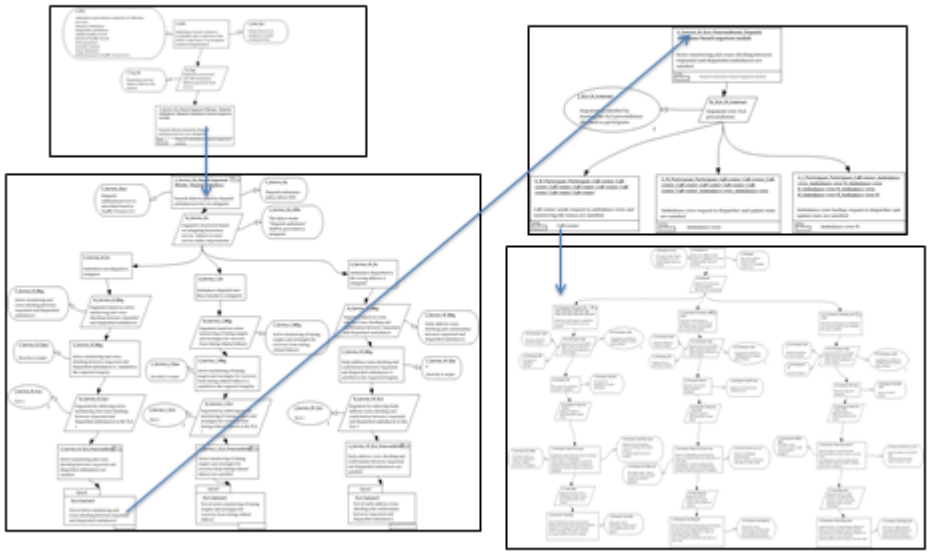
33

Participant Argument



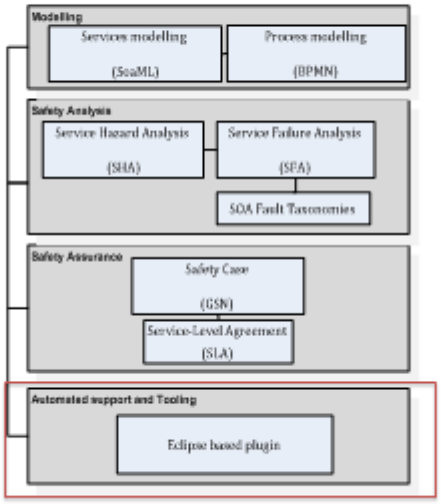
34

Argument Modules for Dispatch Ambulance Service



35

Approach Overview



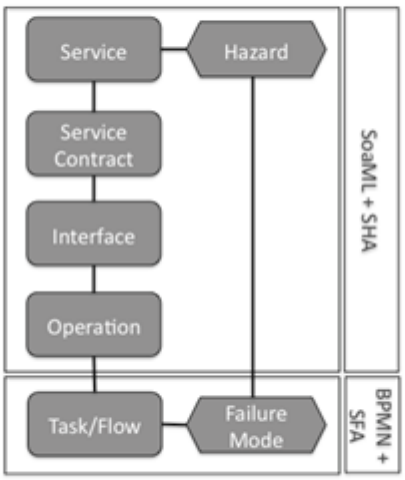
36

Tool-Support

- Create SoaML and BPMN models
- Create and integrate SLAs into SoaML models
- Integrate BPMN and SoaML models
- Embed SHA results into SoaML models
- Embed SFA results into BPMN models
- Create GSN-based arguments and patterns
- Create weaving models to integrate the SOA modelling, safety analysis and safety cases (on-going)

37

Traceability between SoaML and BPMN Models



38

Evaluation

Two case studies:

- Healthcare
  - Three services
    - Ambulance
    - Electronic health records
    - Childbirth
- Industrial Natural gas processing
  - Four services
    - Separate gases
    - Absorb water
    - Heat exchange
    - Separate gas from liquid

39

Conclusions: SOA Modelling

+ Both SoaML and BPMN representation need to be considered in order to justify the safety of resulting design

+ Integrated modelling and analysis offers greater transparency as to how models and analyses are developed

- Lack of coverage of all SoaML design diagrams (e.g. capabilities) which might reveal further contributions to safety

- The use of execution of BPMN models was not fully utilised

40

Conclusions: SOA Safety Analysis

+ SHA and SFA provide a systematic examination of the hazards posed by services and how the detailed failure behaviour of these services can contribute to hazards

+ SHA and SFA offer explicit means for generating service safety requirements

- Potentially generating too many SFA tables with little automated support

- Lack of integration between failure modes and the types of suitable design tactics

41

Conclusions: SOA Safety Cases

+ Safety case patterns were traceable, at the model-level, to sources of information (SoaML, BPMN, SHA and SFA)

+ Safety case structure mirrored the design and improved the clarity of the reasoning

+ Separation of concern enforced via argument contracts based on SLAs (concerns about hazards vs concerns about failures)

- Lack of automated support for argument instantiation from models and analyses (still on going)

42

Further Work

- Integration of safety analysis with search-based technologies (e.g. simulation and model-checking)
- Further tool-Support to implement model-based assurance cases for SOA based on model weaving
- Extending the work to cover runtime composition and certification of SOA

43

Service-Oriented Architectures for Safety Critical Systems

HISE Research Group
Abdulaziz Al-Humam
Supervisor: Ibrahim Habli

June 2015

1

Outline

- Motivation
- Research Hypothesis and Contributions
  - SOA Modelling
  - SOA Safety Analysis
  - SOA Safety Cases
- Evaluation
- Conclusions

2

Background

- Most organisations are based on providing and consuming services
- Most well-known service-based sectors include:
  - Healthcare
  - Banking
  - Retail
  - Transport
  - Emergency
- Services in high-risk sectors can make a direct contribution to safety
  - i.e. failure of services can lead to harm

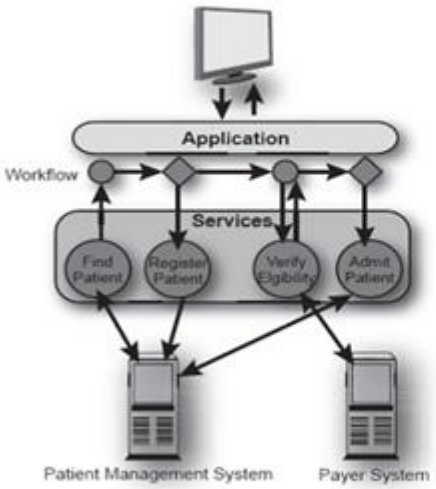
3

Service-oriented Architectures (SOA)

- A **service** is a value delivered to another through a well-defined interface and available to a community
- **SOA** is as a paradigm for defining how people, organisations, and systems provide and use services to achieve results
  - Tasks and data (information flows) are exposed in the form of services
  - Services are used by many applications

4

Example SOA



A conceptual view of healthcare services

Girish Juneja, Blake Dournaee, Joe Natoli & Steve Birkel, 2009

5

Motivation

- To integrate safety analysis and assurance into the engineering of systems designed based on SOA
  - Shortage of research on use of SOAs for development of safety-critical systems
  - Lack of systematic safety analysis processes and methods that address the special features of SOAs
  - Lack of safety assurance strategies for justifying the acceptably safety of SOA

6

Research Hypothesis

Integrated architectural modelling and analysis of safety-critical service-oriented systems provides a **traceable**, **consistent** and **systematic** means for **assuring** the safety of these systems.

7

Research Hypothesis

Integrated architectural modelling and analysis of safety-critical service-oriented systems provides a **traceable**, **consistent** and **systematic** means for **assuring** the safety of these systems.

- **Integrated:** safety analysis and design decisions evolve together
- **Traceable:** forward and backward model-based traceability between design and safety analysis
- **Consistent:** design features are reflected in design and vice versa
- **Systematic:** methodical steps for performing safety assurance
- **Assuring:** explicit reasoning about the safety properties and behaviour

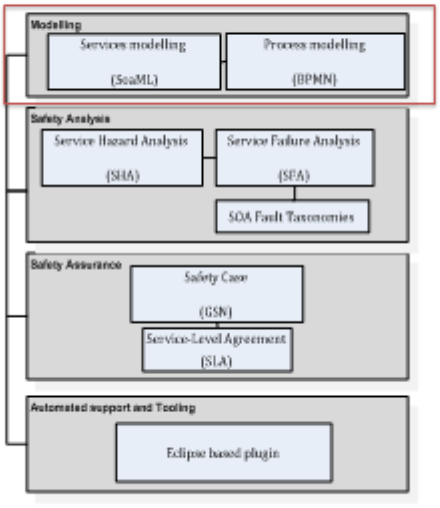
8

Research Contributions

- **Safety Analysis for SOA**
  - Method for analysing the structural view of SOA
    - Service Hazard Analysis (SHA)
  - Method for analysing the behavioural view of SOA
    - Service Failure Analysis (SFA)
- **Safety Case Development for SOA**
  - Modular safety argument patterns for reasoning about SOA safety
- **Automated tool-support**
  - Implementing model-based integration

9

Approach Overview



10

SOA Modelling

Structure and behaviour of SOA represented in two modelling languages:

- Service oriented architecture Modeling Language **(SoaML)**
  - Representation of high-level structure of SOA design
  - Provides ability to specify and analyse requirements for each service
- Business Process Modelling Notation **(BPMN)**
  - Captures the dynamic behaviour implemented in the SOA design
  - Provides the capability to communicate and represent the interaction between detailed service tasks

11

Natural Gas Processing Case Study

Exploratory case study based on sevens services:

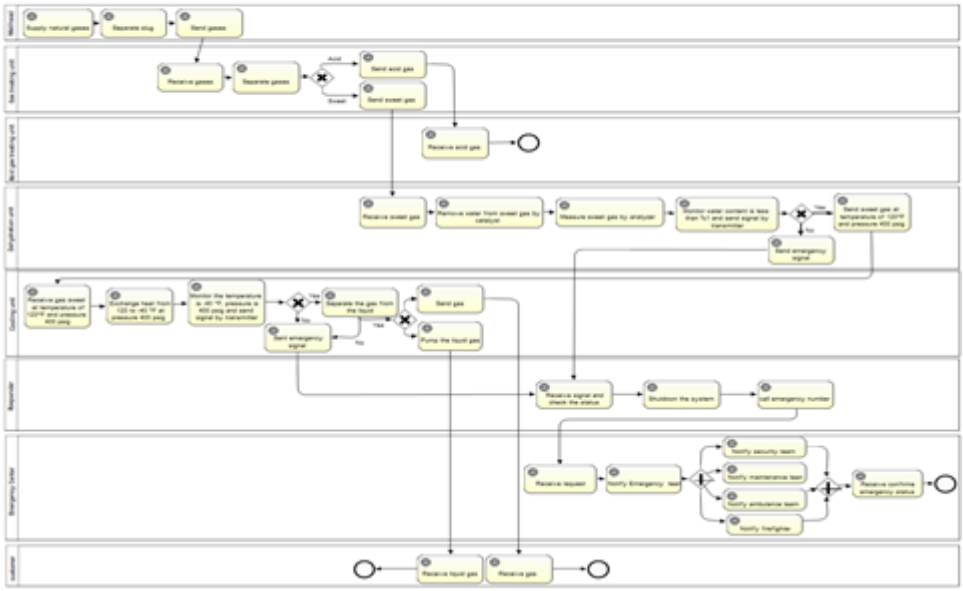
- Supply Natural Gases
- Separate Gases
- Absorb Water
- Heat Exchange
- Separate Gas from Liquid
- Send Emergency Signal
- Call Emergency Number

12

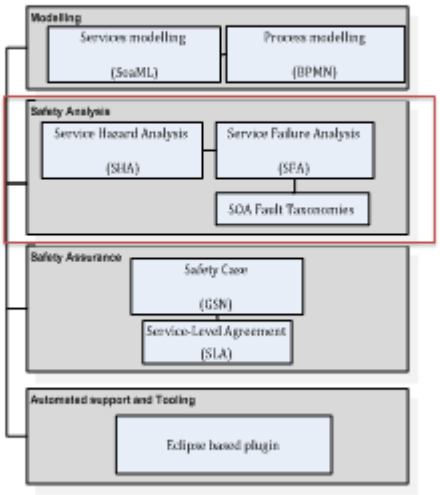
Natural Gas Processing Case Study: SoaML Services Model

13

Natural Gas Processing Case Study: BPMN Model



Approach Overview



15

SOA Safety Analysis

- Method for analysing the structural view of SOA
  - Service Hazard Analysis (SHA)
- Method for analysing the behavioural view of SOA
  - Service Failure Analysis (SFA)
  - SOA Fault Taxonomies in SFA
    - Causes of each of the deviations is linked to a specific type of fault defined in SOA fault taxonomy

16

Service Hazard Analysis (SHA)

- A variant of Functional Hazard Assessment (FHA), called Service Hazard Analysis (SHA)

- Analysis based on three potential service deviations
  - (1) Service not provided when required
  - (2) Service provided when not required
  - (3) Incorrect service

Output of SHA is a set of safety requirements defined at the service level

17

SHA Steps

1. Identify a service
2. Identify the service failure modes
3. Determine the safety effects of each service failure mode
4. Determine the safety severity/classification of each service failure mode
5. Provide service safety requirements
6. Identify potential mitigation measures for meeting the service safety requirements

Consequence Categories

	Worker Safety	Public Safety	Environment	Economic (Annual)
Low, 1	Reportable or equivalent.	None.	Limited impact that is readily corrected	$10.000 to $100.00
Moderate, 2	Hospitalization or lost-time injury.	Minor medical attention.	Report to agencies and take remedial action.	$100.000 to $1 million
Severe, 3	Single disabling.	Hospitalization or serious injury. Some local reporting.	Irreversible damage to low-quality land. Or clean-up of environmentally sensitive areas requires	$1 million to $ 10 million
Very severe, 4	Fatality or multiple serious injuries.	Fatality or multiple serious injuries. Massive negative publicity.	Months of clean-up work needed in environmentally sensitive areas	≥ $10 million

Case Study: SHA Results

Service	Failure Modes	Effects	Severity	SSRs	Mitigation
Absorb water Context: Absorption service means removing the water in gas dehydration processes	Water not removed (i.e. water content is more than 1%)	Blockage in the pipeline Damaged pipelines Explosion	(Very Severe, 4)	The failure mode "water not removed" shall be prevented or mitigated.	Active monitoring and cross-checking between removing water and measure sweet gas through the analyser. Also, checking the catalyst inside the dehydration drum to ensure it is within its service life e.g. needs to be replaced every 5 years.
	Water removed when not required	N/A	N/A	None.	None.
	Water removed later than intended	Freeze inside the pipeline Blockage in the pipeline Damaged pipelines Explosion	(Very Severe, 4)	The failure mode "water removed later than intended" shall be prevented or mitigated.	Active monitoring of timing targets and strategies for recovery from timing-related failures through analyser.
	Water removed earlier than intended	N/A	N/A	None.	None.
	Inaccurate percentage of water removed (i.e. water content is less than 1%)	N/A No direct safety effects The system can handle that without any effect	N/A	None.	None.

Case Study: SHA Results

Service	Failure Modes	Effects	Severity	SSRs	Mitigation
Heat exchange	Gas not cooling	Gas leak within pipelines	(Very Severe, 4)	The failure mode "gas not cooling" shall be prevented or mitigated.	Active monitoring and cross-checking the transmitter to control the liquid level in the cooling by setting point in the system at -40 °F and pressure is 400 psig
	Gas cooling when not required	N/A	N/A	None.	None.
	Gas cooling later than intended	N/A	N/A	None.	None.
	Gas cooling earlier than intended	N/A	N/A	None.	None.
	Inaccurate percentage of gas cooling	Gas leak within pipelines since no gas is passing through cooler	(Very Severe, 4)	The failure mode "inaccurate percentage of gas cooling" shall be prevented or mitigated.	Early temperature cross-checking of the transmitter and confirmation the setting point in the DCS that the cooling unit is -40 °F and pressure is 400 psig

Case Study: SHA Results

Service	Failure Modes	Effects	Severity	SSRs	Mitigation
Send emergence signal	Signal not sent	Damaged pipeline Explosion Fire	(Very Severe, 4)	The failure mode "signal not send" shall be prevented or mitigated.	Use of redundancy by transmitter to confirm the signal has been received
	Signal sent when not required	N/A	N/A	None.	None.
	Signal sent later than intended	Damaged pipeline Explosion Fire	(Very Severe, 4)	The failure mode "signal send later than intended" shall be prevented or mitigated.	Active monitoring of timing targets and strategies for recovery from timing-related failures.
	Signal sent earlier than intended	N/A	N/A	None.	None.
	Inaccurate signal sent (Wrong respond) (Wrong reading)	Damaged pipeline Explosion Fire	(Very Severe, 4)	The failure mode "inaccurate signal send" shall be prevented or mitigated.	Early cross-checking by maintenance technician to confirm that correct the signal has been received.

22

Service Failure Analysis (SFA)

- Analysing the flow of information between interacting tasks

- SHARD safety analysis technique is adapted to analyse the flow of information between the tasks represented
- The output of this analysis is a set of Derived Service Safety Requirements (DSSRs)

23

SFA Steps

1. Identify a flow between two tasks
2. Identify flow failure modes
3. Determine the potential causes of each flow failure mode
4. Determine the potential effects of each flow failure mode
5. Define DSSRs
6. Identify potential mitigation measures for meeting the DSSRs

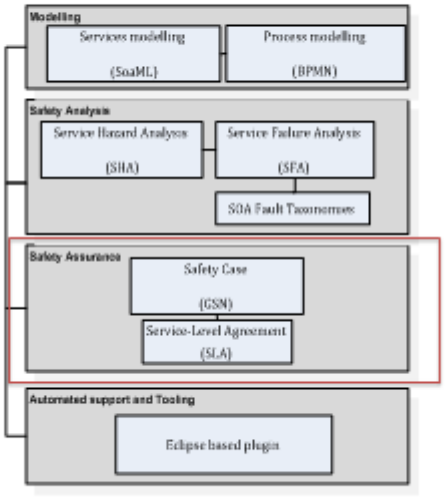
24

Case Study: SFA Results

ID (Flow)	Guide Word	Deviation	Possible Causes	Effects	DSSRs	Mitigation
1.1	Omission (no communication within the signal)	No output signal is sent	Conversation fault (Incorrect input) (server crashed)	No request received	The failure mode "no output signal is sent" shall be prevented or mitigated.	Use of redundancy in sending sources
	Omission (Incomplete signal)	Part of output signal is missing	Required resource missing Service/server incompatible	Inconsistent signal	The failure mode "part of output signal is missing" shall be prevented or mitigated.	Checking and monitoring APIs function and fault containment
1.2	Commission (Repetition)	Additional output signal sent	Redundant description (Incorrect input) (server crashed)	Conflicting request received Wrong responduse Wrong reading signal	The failure mode "with additional output signal sent" shall be prevented or mitigated.	Use high-integrity components among participants to consider sufficiently incorrect signal
	Commission (Spurious)	The communication is unintended	conversation fault (Incorrect input) (server crashed)	Only effected when the message is valid	The failure mode "the communication is unintended" shall be prevented or mitigated.	Use high-integrity components among participants to consider sufficiently incorrect signal
1.3	Early	N/A	N/A	N/A	N/A	Just: the output cannot be produced
1.4	Late	Unacceptable response time to change in demands	Execution fault Time out (severe crash or communication failure)	Overloading signal	The failure mode "Unacceptable response time to change in demands" shall be prevented or mitigated.	Active monitoring by transmitter of timing targets and strategies for recovery from timing-related failures
1.5	Value	One component of output is not proportional to demand	Incompatible components (composition fault)	False signal Hard to detect from flow wrong advice through radio communication channel as well	The failure mode "One component of output is not proportional to demand" shall be prevented or mitigated.	Applying diversity in the flow and fault containment to detect and reject incorrect signal

25

Approach Overview



26

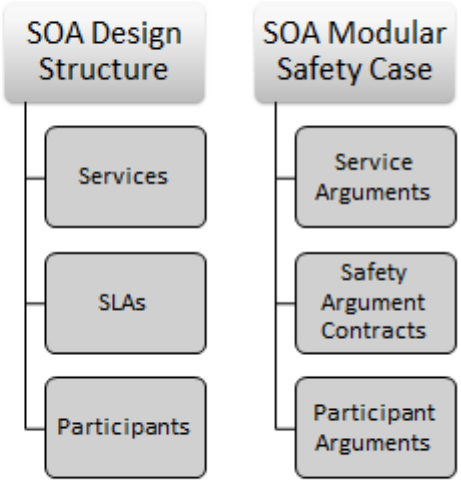
SOA Safety Cases

- Assuring services and their design in SOA safety cases
  - Safety argument is structured based on justification of individual services and their interactions through participants
    - Interaction influenced by pre-defined Service-Level Agreements (SLAs)
- Development of modular safety case patterns for SOA
  - SOA argument patterns catalogue includes rules for pattern composition and traceability to the architectural design (i.e. SoaML and BPMN models) and analysis (i.e. SHA and SFA)

27

SOA and Modular Safety Cases

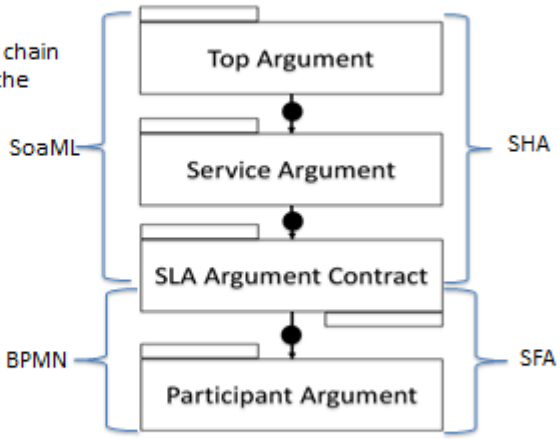
- Correspondence between the high-level SOA design and the overall SOA safety case
- The mapping between SLAs and the argument contracts drives the overall structure of the SOA safety argument and offers traceability between the design and the safety assurance



28

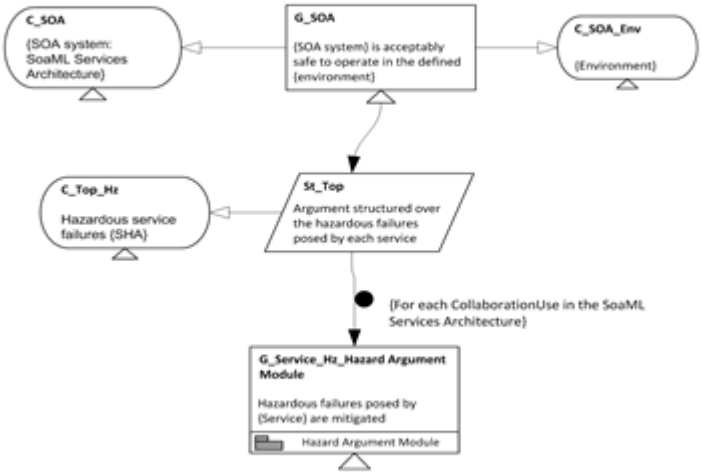
SOA Argument Patterns Catalogue

SOA safety argument pattern catalogue follows a reasoning chain from the Top Argument until the Participant Argument



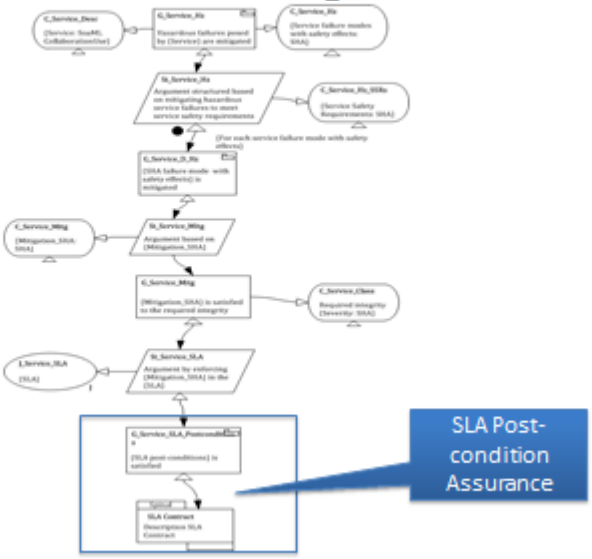
29

Top Argument



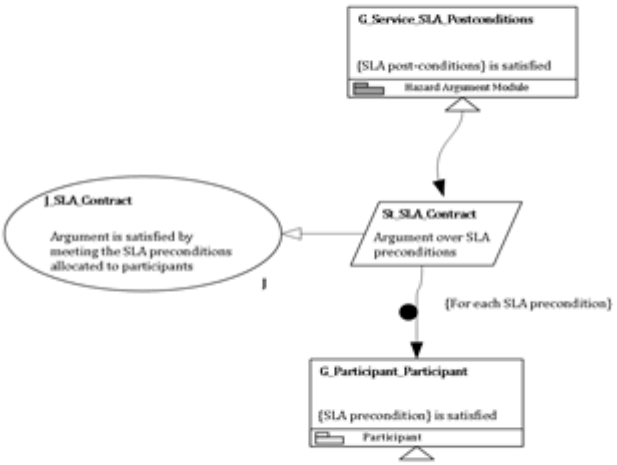
30

Service Hazard Argument



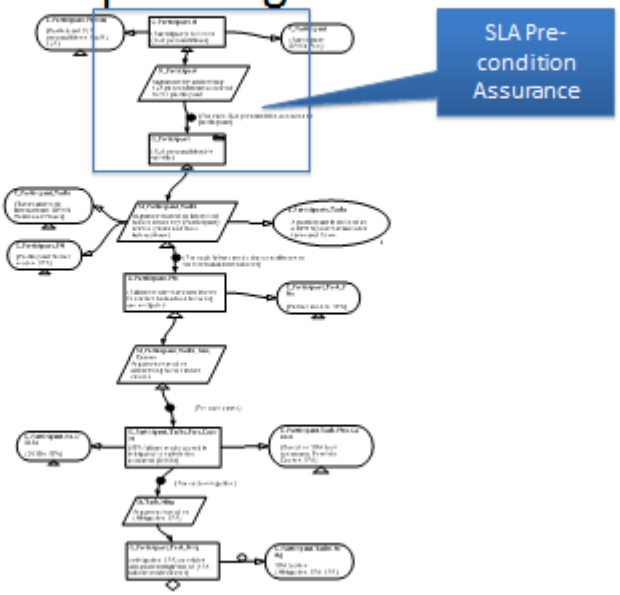
31

SLA Contract Argument



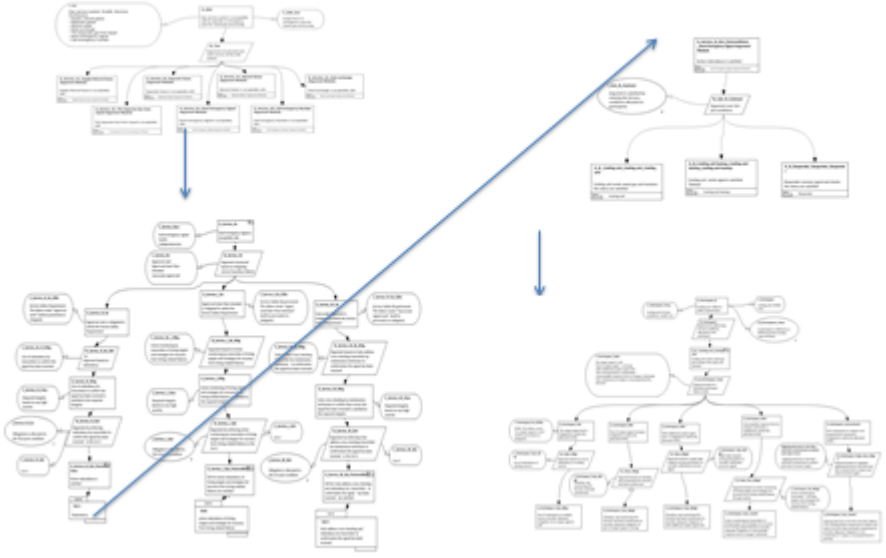
32

Participant Argument



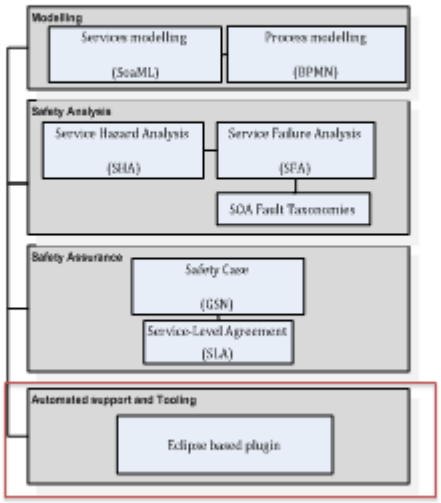
33

Argument Modules for Send Emergency Signal Service



34

Approach Overview



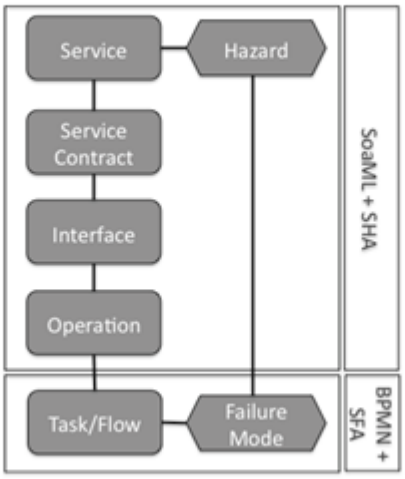
35

Tool-Support

- Create SoaML and BPMN models
- Create and integrate SLAs into SoaML models
- Integrate BPMN and SoaML models
- Embed SHA results into SoaML models
- Embed SFA results into BPMN models
- Create GSN-based arguments and patterns
- Create weaving models to integrate the SOA modelling, safety analysis and safety cases (on-going)

36

Traceability between SoaML and BPMN Models



37

Evaluation

Two case studies:

- Healthcare
  - Three services
    - Ambulance
    - Electronic health records
    - Childbirth
- Industrial Natural gas processing
  - Four services
    - Separate gases
    - Absorb water
    - Heat exchange
    - Separate gas from liquid

38

Conclusions: SOA Modelling

+ Both SoaML and BPMN representation need to be considered in order to justify the safety of resulting design

+ Integrated modelling and analysis offers greater transparency as to how models and analyses are developed

- Lack of coverage of all SoaML design diagrams (e.g. capabilities) which might reveal further contributions to safety

- The use of execution of BPMN models was not fully utilised

39

Conclusions: SOA Safety Analysis

+ SHA and SFA provide a systematic examination of the hazards posed by services and how the detailed failure behaviour of these services can contribute to hazards

+ SHA and SFA offer explicit means for generating service safety requirements

- Potentially generating too many SFA tables with little automated support

- Lack of integration between failure modes and the types of suitable design tactics

40

Conclusions: SOA Safety Cases

+ Safety case patterns were traceable, at the model-level, to sources of information (SoaML, BPMN, SHA and SFA)

+ Safety case structure mirrored the design and improved the clarity of the reasoning

+ Separation of concern enforced via argument contracts based on SLAs (concerns about hazards vs concerns about failures)

- Lack of automated support for argument instantiation from models and analyses (still on going)

41

Further Work

- Integration of safety analysis with search-based technologies (e.g. simulation and model-checking)
- Further tool-Support to implement model-based assurance cases for SOA based on model weaving
- Extending the work to cover runtime composition and certification of SOA

42

Date: 03/09/2015 Areas of expertise of the interviewee: Chemical engineers in the oil and gas industry
1- What do you think about the overall approach? *I can say the overall approach can be very useful to be used in service-based systems.*
2- Do you think that the modelling approach improve the understanding of the system? *Of course, Yes I think it helps making the system structure and behaviour related hazards clearer and more visible through following the logical and layered approach.*
a. What aspects in particular did you find useful? *I can say that it could be the system design models, the systematic procedure of the analysis and also make it appealing and easy to use. I recommend this kind of approach to be utilized.*
b. What aspects in particular did you find hard to use? *Most likely the way it requires to be configured, the tool and the requirements of entering data manually, this area of improvement that needs to be look at carefully. Also, the manual part is the most difficult that needs to be looked carefully as this can reduce the usability and usefulness of the tool.*
3- Do you think that the safety analysis methods, SHA and SFA, can provide a systematic approach to analysing services? *Yes, based on what I have seen,* *Yes, they do help the safety engineers through using this easy to follow up steps, clear methodology or process. We can enhance the engineers or the process developers to follow and track easy.*
a. What aspects do you think have the potential to improve practice? *I will think the integration modelling safety analysis and safety case can help a lot to improve the safety management. This pat need to be looked in it carefully to added value to it.*
b. What aspects do you think are infeasible to use?

I am not sure that I have not apply the approach extensively *So, this require more research, development and digging more to finalize this infeasible area that cannot be use.*
4- Do you think that the safety case used improve the justification for the safety of the service-based system? *Yes, what we have seen and looked in it* *Yes, using safety case patterns can help a lot to establish more structure and logical safety case*
a. What aspects do you think have the potential to improve practice? *Applying what do you call it SHA, SFA and using their result together with safety case patterns* *I think can help and improve the practice as well*
b. What aspects do you think are infeasible to use? *again, Not very sure* *As I have not had the chance to apply the approach thoroughly and extensively* *I think, it require more research to be finalize*
5- Do you think that the modelling, safety analysis and safety case artefacts are clearly integrated? *Yes, what I have seen and looked it,* *I think, Yes, This kind from overall system architecture, it is very clear how the aspects are related and integrated*
6- Can you provide suggestions for areas that can improve the framework? *I think it would be better, if you implement and integrate the model – based weaving approach in the framework. That is added value.*

Table 45: Interview Questions and answer one

Date: 03/09/2015

Areas of expertise of the interviewee: Chemical engineers in the oil and gas industry and overall in processing requirements

1- What do you think about the overall approach?

The overall approach of the system, I can say that is very useful and can help be use in any service-based system with oil and gas domain industry. The overall of the approach is very great.

2- Do you think that the modelling approach improve the understanding of the system?

For the demonstration, I strongly believe that the system structure and behaviour related hazards and failure modes in the system.

So, I believe the system is very clear, friendly to be used and can help a lot meeting overall system objective requirements.

a. What aspects in particular did you find useful?

Typically, for the analysis models and system design models, I think the system procedure of the analysis method make it more applicable and easy to be used

b. What aspects in particular did you find hard to use?

The system design and analysis is very friendly, handy and can be easily understand but I believe that the need for human interface in order to go and enter the data manually can somehow, it will not be an accurate data and infeasible. This in the only area need to be looked in it

3- Do you think that the safety analysis methods, SHA and SFA, can provide a systematic approach to analysing services?

Yes, this will help the engineers overall to use the system, the systematic is very easy to be followed steps are very clear and handy

a. What aspects do you think have the potential to improve practice?

For new system, we need to analysis the system and the safety case and also the integration model. I believe that will help improve the overall system management. This aspect can help improve the practice

b. What aspects do you think are infeasible to use?

To be honest, I start with you from the beginning of the system and I have already applied the approaches in one area of processing oil and gas industry. I see is very useful but I think I need to dig more in order to get the outcome what is an infeasible area.
4- Do you think that the safety case used improve the justification for the safety of the service-based system? *I strongly believe that the safety case can be used to improve* *Definitely, this will help in establish more structure and logical safety case* *Overall, having safety case patterns for overall the system will improve the justification*
a. What aspects do you think have the potential to improve practice? *Looking as the safety analysis methods Basically, SHA and SFA using their result together with safety case. I think this will help improve the practice of the system*
b. What aspects do you think are infeasible to use? *As I highlighted earlier, it is area of more research and when we dig more and more practice. The infeasible will appear and can figure out*
5- Do you think that the modelling, safety analysis and safety case artefacts are clearly integrated? *For sure, it is very clear for the framework that are integrated.*
6- Can you provide suggestions for areas that can improve the framework? *I think, If you can improve and integrated the mode –based weaving* *This is definitely can help to approach the overall framework*

Table 46: Interview Questions and answer two

Date: 26 August 2015

Areas of expertise of the interviewee: Senior Safety Engineering (Healthcare domain)

1- What do you think about the overall approach?

I can see a lot of applications, especially in my domain, healthcare.

The modelling approach produces a strong basis for the risk assessment

2- Do you think that the modelling approach improve the understanding of the system?

Focusing on services gives an easier way to manage and visualise the systems for non-experts.

Services provide a clearer way of presenting the system as compared to functions.

a. What aspects in particular did you find useful?

Producers and consumers and the relation between them (dependencies between people and systems).

Richer than a textual narrative for complex situations.

Better than more complex models such as state machines.

b. What aspects in particular did you find hard to use?

Initial period for training and learning the language (but not for engineers).

Not difficult for engineers because of the limited number of elements to model.

3- Do you think that the safety analysis methods, SHA and SFA, can provide a systematic approach to analysing services?

Yes, especially that they are built on familiar safety techniques.

a. What aspects do you think have the potential to improve practice?

Not much a new element of re-learning.

Repeatable.

Keeps the relation between the hazards, services and tasks clear.
b. What aspects do you think are infeasible to use? *Language control: different people have different styles or use different terminology: faults vs failures (hard to trace and map different expressions of the same condition)*
4- Do you think that the safety case used improve the justification for the safety of the service-based system? *The modular view provides the right level of abstraction* *The modular view provides value in showing the big picture.*
a. What aspects do you think have the potential to improve practice? *The link between the hazards and causes* *Traceability between the hazards and the evidence*
b. What aspects do you think are infeasible to use? *Detailed arguments (detailed pattern instantiation).* *People are not trained to do it/perceived to take a lot of time.*
5- Do you think that the modelling, safety analysis and safety case artefacts are clearly integrated? *Yes, without a model you cannot do the analysis.* *I understand how they interrelate from the framework.*
6- Can you provide suggestions for areas that can improve the framework? *Make sure to maintain a thread between the hazards and failures.* *How much enough analysis is enough? How much details are needed? When do we stop at the SLA level? (abstract how participants do)*

Table 47: Interview Questions and answer three

Appendix G. Weaving Model

A preliminary weaving model, based on the work in (164), shown in Figure 72, is used for defining the dependencies, mainly at the metamodel level between, on the one hand, the SOA safety argument patterns and, on the other hand the SoaML, BPMN, SLAs, SHA and SFA metamodels. This aims at improving traceability and allowing the argument pattern instantiation to be automatically executed. The instantiation program, developed in (164), runs on the Eclipse platform. It identifies the required information from those information models (SoaML and BPMN) and performs the instantiation. Overall, the tool-support in this thesis offered a means for checking the consistency and well-formedness of the models and metamodels defined in the SOA safety assurance framework.

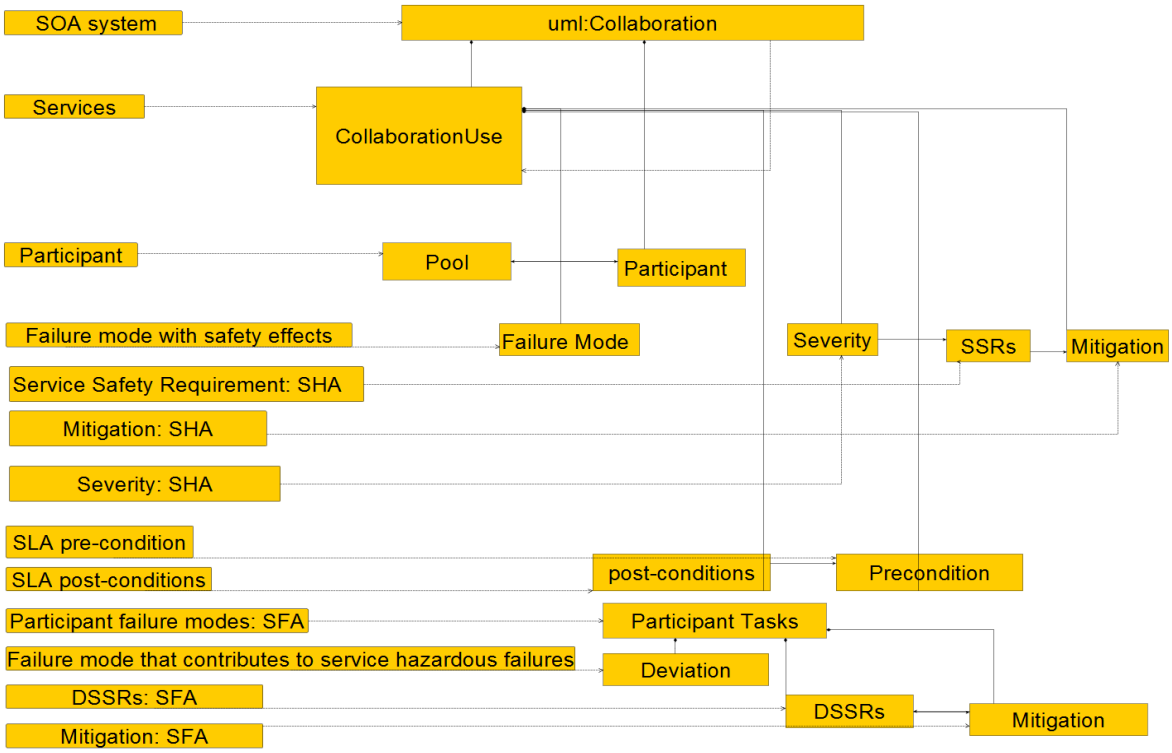


Figure 72: Preliminary Weaving Model

Appendix H. Abbreviations and Acronyms

SOA	Service-Oriented Architecture
BPMN	Business Process Management Notation
SoaML	Service oriented architecture Modeling Language
AADL	Architecture Analysis and Design Language
COTS	Commercial Off-The-Shelf
EHR	Electronic Health Record
GSN	Goal Structuring Notation
HAZOP	Hazard and Operability Studies
HISE	High-Integrity Systems Engineering
MoD	Ministry of Defence (UK)
EMR	Electronic Medical Record
FFA	Functional Failure Analysis
FHA	Functional Hazard Analysis
FMEA	Failure Mode Effect Analysis
FTA	Fault Tree Analysis
NHS	National Health Service
OMG	Object Management Group
SHARD	Software Hazard Analysis and Resolution in Design
SFA	Service Failure Analysis
SHA	Service Hazard Analysis
UML	Unified Modelling Language
SysML	Systems Modelling Language
SLA	Service Level Agreement
NASA	National Aeronautics and Space Administration
ESA	European Space Agency
OASIS	Organization for the Advancement of Structured Information Standards
QoS	Quality of Service
EIA	Enterprise Integration Applications

ROI	Return-On-Investment
TMR	Triple Modular Redundancy
SDL	Specification and Description Language
SOMA	Service-Oriented Modelling and Architecture
SOMF	Service-Oriented Modelling Framework
BPMI	Business Process Management Initiative
MA	Markov Analysis
FPTC	Fault Propagation and Transformation Calculus
PSSA	Preliminary System Safety Assessment
ATAM	Architecture Tradeoff Analysis Method
IEC SC 54A	International Electro-technical Commission subcommittee 54A
IMA	Integrated Modular Avionics
SAAM	Software Architectural Analysis Method
DGR	Dependency-Guarantee Relationships
IDE	Integrated Development Environment
BPM	Business Process Management
SSRs	Service Safety Requirements
DSSRs	Derived Service Safety Requirements
EHR	Electronic Health Record
CO2	Carbon Dioxide
H2S	Yydrogen Sulphide
EG	Ethylene Glycol
DEG	Diethylene Glycol
TEG	Triethylene Glycol
TREG	Tetraethylene Glycol
SRAM	Safety Report Assessment Manual

Bibliography

1. **Chen, Y.** Service Oriented Architecture. *IBM.* [Online] 2006. [Cited: 12 Apr. 2011.] http://sei.pku.edu.cn/~huanggang/ibmcourse/2006/SOA.pdf.

2. **Object Management Group.** *Service oriented architecture Modeling Language (SoaML) Specification,.* s.l. : Version 1.0.1 (May 2012)., 2012.

3. **Erl, T.** *Service-oriented architecture: concepts, technology, and design.* United States. : Prentice Hall Professional Technical Reference, 2005. ISBN: 2005925019.

4. **Josuttis, J. M.** *SOA in practice.* USA : O'Reilly Media, Inc., 2007. ISBN: 9780596529550.

5. **Erl, Thomas.** *SOA Design Patterns.* United States : Prentice Hall, 2009. ISBN: 9780136135166.

6. **OASIS.** [Online] [Cited: 18 05 2012.] http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=soa-rm.

7. **Open Group.** [Online] [Cited: 2012 05 18.] https://collaboration.opengroup.org/projects/soa/pages.php?gpid=442&action=show&ggid=1574.

8. **Endrei, M., et al., et al.** Patterns: Service-Oriented Architecture and Web Services . *IBM website.* [Online] 2004. [Cited: 6 May 2011.] http://www.redbooks.ibm.com/redbooks/SG246303/wwhelp/wwhimpl/java/html/wwhelp.htm.

9. **Mohiuddin, S. A.** The Business-Technology Drivers and Benefits of SOA. *BTTrends.* [Online] 2007. [Cited: 1 May 2011.] http://www.bptrends.com/publicationfiles/EIGHT%20BPTrends BusTechDrivers%26BenfOfSOA-TelcoExecOverview-Final.pdf.

10. **Glen, S. M. and Andexer, J.** A practical application of SOACombine the technology and business perspectives of SOA implementation. *IBM.* [Online] 4 Oct. 2007. [Cited: 23 Apr. 2011.] https://www.ibm.com/developerworks/webservices/library/ws-soa-practical/.

11. **Juneja, G,, et al., et al.** Improving Performance of Healthcare Systems with Service Oriented Architecture. *infoq.* [Online] 2008 . http://www.infoq.com/articles/soa-healthcare.

12. **Leveson, N. G.** *Software system safety and computers.* USA. : Addison Wesley, 1995. ISBN: 0-201-11972-2.

13. *Design Factors for Safety-Critical Software,.* **Lawrence, J.D.** 1994, NUREG/CR-6294, Lawrence NUREG/CR-6294, Lawrence.

14. **Kelly, T.** *Arguing Safety - A Systematic Approach to Safety Case Management.* York : Department of Computer Science, University of York, 1998. DPhil Thesis.

15. **Fenn, J,, et al., et al.** *Safety case composition using contracts – refinements based on feedback from an industrial case study,.* s.l. : 15th Safety-Critical Systems Symposium (Bristol, UK,), 2007.

16. **Brown, A., Fenn, J. and Menon, C.** Issues and considerations for a modular safety certification approach in a Service-Oriented Architecture . *In 5th IET International System Safety Conference.* 2010, pp. 1 - 6 .

17. **Roshen, W.** *SOA-based enterprise integration: a step-by-step guide to services-based application integration.* USA : McGraw Hill Professional, 2009. ISBN: 9780071605526.

18. **Krafzig, D., Banke, K. and Slama, D.** *Enterprise SOA: service-oriented architecture best practices.* USA : Prentice Hall PTR, 2005. ISBN: 9780131465756.

19. **Güner, S.** *Architectural Approaches, Concepts and Methodologies of Service Oriented Architecture.* Hamburg : Technical University Hamburg Harburg, 2005.

20. **McGovern, J., et al., et al.** *Java Web Services Architecture.* s.l. : Sun Microsystems, 2003. ISBN: 978-1558609006.

21. **Silva, P. A., et al., et al.** Extending SOA with Semantic Mediators. *In Advanced Internet Based Systems and Applications, Lecture Notes In Computer Science, Springer-Verlag, Berlin, Heidelberg.* 2009, Vol. 4879, pp. 362-371.

22. **Jazequel, J.-M. and Meyer, B.** Design by contract : the lessons of Ariane. 1997, Vol. 30, 1.

23. **Cellary, W. and Strykowski, S.** *e-government based on cloud computing and service-oriented architecture.* New York, NY, USA : ACM, 2009.

24. **Valipour, M.H., et al., et al.** A brief survey of software architecture concepts and service oriented architecture. Beijing : IEEE, 2009.

25. **Alexander, R.** *Using Simulation for Systems of Systems Hazard Analysis.* s.l. : PhD thesis, Department of Computer Science, University of York., 2007.

26. **Hawkins, R. D.** *Using Safety Contracts in the Development of Safety Critical Object-Oriented Systems.* York : Department of Computer Science, University of York, 2006.

27. **Habli, I.** *Model-Based Assurance of Safety-Critical Product Lines, PhD Thesis.* York : Department of Computer Science, University of York, 2009.

28. **Higgins, John.** *ITIL and the Software Lifecycle: Practical Strategy and Design Principles.* s.l. : Van Haren Publishing, 2007. ISBN: 9789087530495.

29. **Seeley, R.** Understanding the SOA lifecycle. *Search SOA.* [Online] 24 Dec. 2007. [Cited: 6 May 2011.] http://searchsoa.techtarget.com/tip/Understanding-the-SOA-lifecycle.

30. **Cox, D. E. and Kreger, H.** Management of the service-oriented-architecture life cycle. *IBM System Journal,.* 2005, Vol. 44, 4, pp. 709-726. http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=5386697&abstractAccess=no&userType=inst.

31. **Kumar, S.** *The Enterprise SOA Implementation Lifecycle Explained.* India. : SAP DEVELOPER NETWORK (sdn.sap.com), 2007.

32. **Gejnevall, Mats.** Open Group SOA Governance . *SoA Lifecycle .* [Online] 21 Aug. 2007. [Cited: 8 May 2011.] http://soalifecycle.blogspot.com/.

33. **Ramollari, E., Dranidis, D. and Simons, A.** A Survey of Service Oriented Development Methodologies. *Proc. Second European Young Researchers Workshop on Service Oriented Computing.* 2007.

34. **Morris, E., et al., et al.** *Testing in Service-Oriented Environments.* United States : Carnegie Mellon University, 2010. CMU/SEI-2010-TR-011.

35. **Fronckowiak, J.** SOA Best Practices and Design Patterns: Key to Successful Service Oriented Architecture Implementation. Oracle Corporation, 2008.

36. **Alexander, C., Ishikawa,S.,Silverstein,M,. Jacobson, M., Fiksdahl-King, I. and Angel, S.** *A Pattern Language: Towns, Buildings, Construction.* Oxford : Oxford University Press, 1977. 0195019199.

37. **Gamma, E., Helm, R., Johnson,R. and Vlissides., J.** *Design Patterns: Elements of Reusable Object-Oriented Software .* USA : Addison-Wesley, 1994. ISBN 0201633612.

38. **Spool, J.** The Elements of a Design Pattern. *User Interface Engineering.* [Online] Jan 23, 2006. [Cited: May 02, 2011.] http://www.uie.com/articles/elements_of_a_design_pattern/.

39. **Metsker, S. and Wake, W.** *Design patterns in Java.* USA : Addison-Wesley Professional, 2006. ISBN: 9780321333025.

40. **Cooper., J.** *Java design patterns: a tutorial.* Massachusetts : Addison-Wesley Professional, 2000. ISBN 9780201485394.

41. **Laprie, J.C.** Dependable Computing and Fault Tolerance: Concepts and terminology. *Highlights from Twenty-Five Years, Twenty-Fifth International Symposium on Fault-Tolerant Computing.* 1995.

42. **IFIP.** WG 10.4 on DEPENDABLE COMPUTING AND FAULT TOLERANCE. *INTERNATIONAL FEDERATION FOR INFORMATION PROCESSING.* [Online] 23 Jul. 2010. [Cited: 5 May 2011.] http://www.dependability.org/wg10.4/.

43. **Avizienis, A., Laprie, J.-C. and Randell, B.** *Fundamental Concepts of Dependability.* Boston, Massachusetts, USA : In Proc. 3rd Information Survivability Workshop,LAAS-CNRS, 2001. Research Report No 1145.

44. **Despotou, G.** *Managing the Evolution of Dependability Cases for Systems of Systems.* York, United Kingdom : PhD Thesis, High Integrity Systems Research Group, Department of Computer Science, University of York, 2007.

45. **Avizienis, Algirdas, Laprie, Jean-Claude and Randell, Brian.** Dependability and its threats: a taxonomy. *18th IFIP World Computer Congress, Building the Information Society.* 2004, pp. 91-120.

46. **Bass, Len, Clements, Paul and Kazman, Ric.** *Software architecture in practice.* Boston, United States. : Addison-Wesley, 2003. ISBN: 9780321154958.

47. **Cigital.** Software Certification. *Cigital Software Confidence Achieved.* [Online] 2011. [Cited: 24 May 2011.] http://www.cigital.com/labs/reliability/certification.php.

48. **Chan, C.** Supermarket Accidentally Opens When No Employees Working. *GIZMODO.* [Online] 26 Apr. 2011. [Cited: 23 May 2011.] http://www.gizmodo.com.au/2011/04/supermarket-accidentally-opens-when-no-employees-working/.

49. **Hallstrom, J. O., Soundarajan, N. and Tyler, B.** *Monitoring Design Pattern Contracts.* IOWA : IOWA State University, 2004. 04-09.

50. **Wu, W. and Kelly, T.** Safety Tactics for Software Architecture Design. *In: Proc. 28th Annual International Computer Software and Applications Conference COMPSAC.* 2004, Vol. 1, pp. 368 - 375.

51. **Surhone, Lambert M., Tennoe, Mariam T. and Henssonow, Susan F.** *Triple Modular Redundancy.* s.l. : VDM Verlag Dr. Mueller AG & Co. Kg, 2010. ISBN: 9786132988683.

52. **Jackson, Daniel.** A Direct Path to Dependable Software. *Communications of the ACM .* 2009, Vol. 52, 4.

53. **Kang, E.** *A Framework for Dependability Analysis of Software Systems with Trusted Bases.* Boston : Massachusetts Institute of Technology, 2010. Master Thesis.

54. **Kandukuri, BR,, Paturi, VR, and Rakshit, A.** *Cloud security issues.* s.l. : IEEE internationalconference on services computing, p. 517–20., 2009.

55. **Ludwig, et al., et al.** *Template-Based Automated Service Provisioning - Supporting the Agreement-Driven Service Life-Cycle.* s.l. : In: Proc. Intl. Conf. Service Oriented Computing, ICSOC'05, pp. 283--295, 2005.

56. **Chan, M. K. S., et al., et al.** *A Fault Taxonomy for Web Service Composition.* Vienna, Austria, : In Proceedings of The Workshop on Engineering Service-Oriented Applications: Analysis, Design and Composition (WE-SOA'2007), 2007.

57. **Leopoldi, R.** *IT service management Service-level agreement.* s.l. : Sample Agreements, Reports, and. Checklists, White paper, 2002.

58. **Wu, Weihang.** *Architectural Reasoning for Safety-Critical Software Applications.* York : University of York, 2007. Phd Thesis.

59. **Kolovos, D.** *An Extensible Platform for Specification of Integrated Languages for Model Management.* York : Computer Science Department, University of York, 2008.

60. **Björnander, S, and Grunske, L.** *Architecture Description Languages for Automotive Systems – A Literature Review .* s.l. : Technical Report: Swinburne University of Technology, Australia , 2008.

61. **The United Modelling Language (UML) Specification.** s.l. : Version 2.0 ed. : The Object Management Group. , 2005.

62. **SysML Specification.** *Systems Modeling Language (SysML) Specification v. 1.0.* 2005.

63. **Briol, P.** *BPMN, the Business Process Modeling Notation Pocket Handbook.* s.l. : Lulu.com, 2008. ISBN: 9781409202998.

64. **Arsanjani, A.** Service-oriented modeling and architecture: How to identify, specify, and realize services for your SOA . *IBM website.* [Online] 9 Nov. 2004. [Cited: 6 May 2011.] http://www.ibm.com/developerworks/library/ws-soa-design1/.

65. **Methodologies Corp.** Introduction to SOMF. *Methodologies Corp.* [Online] 2011. [Cited: 22 May 2011.] http://www.modelingconcepts.com/pages/download.htm.

66. **White, S. A.** *Introduction to BPMN.* s.l. : IBM Corporation, 2006. http://www.omg.org/bpmn/Documents/Introduction_to_BPMN.pdf.

67. **Pawar, S.** *BPMN tools---a comparative analysis to improve interoperability.* s.l. : College of Technology Masters Theses, Purdue University, 2011.

68. **OMG.** Documents Associated with Service oriented architecture Modeling Language (SoaML). *OMG website.* [Online] Dec. 2009. [Cited: 6 May 2011.] http://www.omg.org/spec/SoaML/1.0/Beta2/.

69. **MINERVA.** eclipse. *eclipse.* [Online] n.d. http://alarcos.esi.uclm.es/MINERVA/TOOLS/soamlPlugin.htm.

70. **Roland, Harold E. and Moriarty, Brian.** *System safety engineering and management.* s.l. : Wiley-IEEE, 1990. 9780471618164.

71. **McDermid, J.A. and Rook, P.** *Software Development Process Models, in Software Engineer's Reference Book.* Butterworth-Heinemann, Oxford : Butterworth-Heinemann Ltd, 1991.

72. **Pumfrey, D. J.** *The Principled Design of Computer System Safety Anaylsis.* York : University of York, 1999.

73. **FAA.** Chapter 8: Safety Analysis/Hazard Analysis Tasks. *FAA System Safety Handbook.* http://www.atcvantage.com/docs/FAA_SystemSafetyHandbook.pdf : FAA, 2000.

74. **Lambert, H. E.** *Use of Fault Tree Analysis for Automotive Reliability and Safety Analysis.* Detroit, Michigan : Lawrence Livermore National Laboratory, 2003. UCRL-JC-154905.

75. **Baltus, B.** *Software quality: state of the art in management, testing, and tools.* Germany : Springer, 2001. ISBN: 9783540414414.

76. **Stamatis, D. H.** *Failure mode and effect analysis: FMEA from theory to execution, 2ed edition.* s.l. : ASQ Quality Press, 2003. ISBN: 9780873895989.

77. **SAE, ARP 4754.** *Guidelines and Methods for Conducting the Safety Assessment Process on Civil Airborne Systems and Equipment.* 1996. Society of Automotive Engineers, Inc, Warrendale, PA..

78. **CISHEC, A .** Guide to Hazard and Operability Studies,. s.l. : The Chemical Industry Safety and Health Council of the Chemical Industries Association Ltd., 1977.

79. **Kletz, T,.** *Hazop and Hazan: Identifying and Assessing Process Industry Hazards.* s.l. : Thirded, Institution of Chemical Engineers. ISBN 0-85295-285-6., 1992.

80. **Hillson, David and Murray-Webster, Ruth.** *Understanding and managing risk attitude.* England : Gower Publishing, Ltd., 2007. ISBN: 9780566087981.

81. **Kasse Initiatives.** Risk Management for Systems Engineering. *Defense Technical Information Centre.* [Online] 2004. [Cited: 19 May 2011.] http://www.dtic.mil/ndia/2004cmmi/CMMIT1Mon/Track1IntrotoSystemsEngineering/KISE09RiskManagementv2.pdf.

82. **Beroggi, Giampiero E. G., Wallace, William A. and Zürich, Eidgenössische Technische Hochschule.** *Computer supported risk management.* Amsterdam. : Springer, 1995. ISBN: 9780792333722.

83. **SAE, ARP 4761.** *Guidelines and Methods for Conducting the Safety Assessment Process on Civil Airborne Systems and Equipment .* Warrendale, PA : Society of Automotive Engineers, Inc., 1996.

84. **Rehn, C.** Software Architectural Tactics and Patterns for Safety and Security . *christian-rehn.* [Online] 3 June 2010. [Cited: 6 May 2011.] http://www.christian-rehn.de/downloads/.

85. **Kazman, R, Klein, M, and Barbacci, M,.** *The Architecture Tradeoff Analysis Method.* s.l. : ICECCS, IEEE CS Press, pp. 68--78., 1998.

86. **Alexander, Robert, Hall-May, Martin and Kelly, Tim.** *Certification of Autonomous Systems.* Edinburgh, : In Proceedings of the 2nd Systems Engineering for Autonomous Systems (SEAS) Defence Technology Centre (DTC) Annual Technical Conference, 2007.

87. **IVAN, I., et al., et al.** *Supervising Software Certification Process.* 2010. http://www.contabilizat.ro/file/cursuri_de_perfectionare/informatica_economica/software_2003/cap23.pdf.

88. **Herrmann, D. S.** *Software safety and Reliability.* California, United States : Angela Burgess, 1999. ISBN: 0-7695-0299-7.

89. **Penny, J., et al., et al.** The practicalities of goal-based safety regulation. *In: Proceedings of the 9th safety-critical systems symposium.* 2001, pp. 35–48 .

90. **Squair, Matthew John.** *Issues in the Application of Software Safety Standards.* Sydney : In 10th Australian Workshop on Safety Related Programmable Systems (SCS'05), 2005.

91. **Hereña, Peter G.** Supplementing Risk-Based Approaches with Prescriptive Templates. s.l. : Kenexis, 2009. TM011.pgh.

92. **Hamilton, V.** Criteria for Safety Evidence: Goal-based standards require evidence-based approaches. *Safety Systems .* 2006, Vol. 16, 1.

93. **Kelly, T., McDermid, J. and Weaver, R.** Goal-Based Safety Standards: Opportunities and Challenges. *in Proceedings of the 23rd International System Safety Conference.* 2005.

94. **Bishop, P.G., Bloomfield, R. and Guerra, S.** *The Future of Goal-Based Assurance Cases.* London : In Proc. ICDSN Workshop on Assurance Cases, 2004. pp. 390–395.

95. **Defence, Ministry of.** *Defence Standard 00-56: Safety Management Requirements for Defence Systems.* UK : Ministry of Defence, 2007. DEF STAN 00-56.

96. **Cockram, T. and Lockwood, B.** *Electronic Safety Case: Challenges and Opportunities.* s.l. : Praxis Critical Systems Limited and Raytheon Systems Limited, 2003.

97. **Goal Structuring Notation standard, (GSN).** [Online] Available at http://www.goalstructuringnotation.info.

98. **Weaver, R., McDermid, J. and Kelly, T.** *Software safety arguments: Towards a systematic categorisation of evidence.* s.l. : In 20th International System Safety Conference, 2002.

99. **Kelly, T. P.** *Concepts and Principles of Composotional Safety Case.* University of York, York. : QinetiQ, 2001. COMSA/2001/1/1.

100. **Bate, I. and Kelly, T.** *Architectural Considerations in the Certification of Modular Systems.* Berlin : Springer-Verlag, 2002. Reliability Engineering and System Safety, vol. 81, Issue 3 Pages 303-324.

101. **Hawkins, R. D. and Kelly, T. P.** *A systematic Approach for Developing Software Safety Arguments.* Huntsville AL, USA : In proceedings of the 27th System Safety Society (SSS) International System Safety Conference (ISSC), 2009.

102. **Kazman, R, et al., et al.** *SAAM: A Method for Analyzing the Properties of Software Architectures.* s.l. : In Proceedings of the 16th International Conference on Software Engineering, Sorrento, Italy, pp. 81–90., 1994.

103. **Hollow, Paul, Mcdermid, John and Nicholson, Mark.** *Approaches to certification of reconfigurable IMA systems.* York. : In Proceedings 10th International Symposium of the International Coucil on Systems Engineering, 2000.

104. **Conmy, P. M.** *Safety Analysis of Computer Resource Management Software.* York : Department of Computer Science, University of York, 2005.

105. **Guerra, P., et al., et al.** *Integrating COTS Software Componentes Into Dependable Software Architectures.* Los Alamitos : In: Proceedings of the 6th International Symposium on Object-Oriented Real-Time Distributed Computing. IEEE Computer Society Press, 2003.

106. **Ye, Fan.** *Justifying the use of COTS components within safety critical applications.* York : Department of Computer Science, University of York, 2005.

107. **Rushby, John.** *Kernels for Safety?* Glasgow, UK. : Blackwell Scientific Publications, 1989. Safe and Secure Computing Systems, pp. 210-220.

108. **Fenn, J., et al., et al.** *The Who, Where, How, Why and When of Modular and Incremental Certification.* London : in proceedings of the 2nd IET International Conference on System Safety, 2007.

109. **Rushby, J.** *CSL Technical Report: Modular Certification.* Menlo Park, USA : Computer Science Laboratory, 2001.

110. **Wilson, A. and Preyssler, T.** *Incremental certification and Integrated Modular Avionics.* UK. : Aerospace and Electronic Systems Magazine, IEEE , vol.24, no.11, pp.10-15, 2009.

111. **R. Donini, S,, et al., et al.** *Testing Complex Safety-Critical Systems in SOA Context.* s.l. : in Complex, Intelligent and Software Intensive Systems, International Conference , 2008. vol., no., pp.87-93, 4-7 March 2008.

112. **Rychlý, M.** *A case study on behavioural modelling of service-oriented architectures.* s.l. : In Software Engineering Techniques In Progress, pp. 79–92. AGH University Press., 2009.

113. —. *A Metamodel for Modelling of Component-Based Systems with Mobile Architecture.* s.l. : Department of Information Systems, Faculty of Information Technology, Brno University of Technology, Boˇzetˇechova 2, 612 66 Brno, Czech Republic, e-mail: rychly@fit.vutbr.cz, 2011.

114. **Rodrigues, D and et al.** *Service-oriented architectures for complex safetycritical embedded systems: A case study on uavs .* s.l. : 1st Brazilian Conference on Critical Embedded Systems (CBSEC), p. 130, , 2011 .

115. **Rodrigues, D and al, et.** *Application of SOA in safety-critical embedded systems.* s.l. : n G. Lee, D. Howard, and D. Ślęzak, editors, Convergence and Hybrid Information Technology, volume 206 of Communications in Computer and Information Science, pages 345--354. Springer Berlin Heidelberg, , 2011.

116. **Monteiro, E and Silva, P.** *A Risk Management Model for Service-Oriented Architecture.* s.l. : International Journal of Advanced Computer Technology (IJACT). Page 53-61. ISSN:2319-7900., 2015.

117. **OMG.** Documents Associated With. *OMG website.* [Online] May. 2012. [Cited: 6 June 2012.] http://www.omg.org/spec/SoaML/1.0.1/PDF.

118. **SAE, ARP4761.** *Guidelines and Methods for Conducting the Safety Assessment Process on Civil Airborne Systems and Equipment.* Warrendale, PA : Society of Automotive Engineers, Inc., 1996.

119. **T. Kletz.** *Hazop and Hazan: Identifying and Assessing Process Industry Hazards,.* s.l. : Institution of Chemical Engineers, Third ed.,, 1992.

120. **Buckley, I., et al., et al.** *Towards pattern-based reliability certification of services.* s.l. : On the Move to Meaningful Internet Systems, Hersonissos, Greece,, 2011.

121. **Bruning, S., Weissleder, S. and Malek, M.** *A Fault Taxonomy for Service-Oriented architecture.* Systems Engineering Symposium, pages 367–368, : In HASE '07: Proceedings of the 10th IEEE High Assurance Systems Engineering Symposium, pages 367–368,IEEE Computer Society., 2007.

122. **Coleman, J., and Jones C.** *A structural proof of the soundness of rely/guarantee rules.* s.l. : Journal of Logic and Computation, 17,, 2007.

123. **Kelly, T.** *Arguing Safety - A Systematic Approach to Safety Case Management.* York : Department of Computer Science, University of York, 1998. DPhil Thesis.

124. eclipse. *eclipse.* [Online] n.d. http://alarcos.esi.uclm.es/MINERVA/TOOLS/soamlPlugin.htm.

125. **papyrus.** papyrus. *papyrus.* [Online] n.d. https://www.eclipse.org/papyrus/.

126. Activiti. *Activiti.* [Online] n.d. http://activiti.org/.

127. **Reason, J.,.** *Human error: models and management.* s.l. : BMJ, 320, 768 (March 2000), 2000.

128. **NHS.** *NHS choices.* s.l. : http://healthguides.mapofmedicine.com/choices/map/index.html, 2011.

129. **The Communications Directorate South West Thames Regional Health Authority, .** *Report of the Inquiry Into The London Ambulance Service.* London : International Workshop on Software Specification and Design Case Study, Original ISBN No: 0 905133 70 6, 1993.

130. **Musick, E.** *The 1992 London Ambulance Service Computer Aided Dispatch System Failure.* s.l. : MSOE - SE-3811 - FORMAL METHODS, DR. WELCH, SPRING, 2006.

131. **Health and Social Care Information Centre.** *Ambulance Services.* England : s.n., 2014.

132. **NHSLA Risk Management Standards for Ambulance Services.** *NHS Litigation Authority.* s.l. : DNV Healthcare UK, 2012.

133. **Finkelstein, A.** *Report of the Inquiry Into The London Ambulance Service.* London) : International Workshop on Software Specification and Design Case Study,University College London, 1993.

134. **Welsh Ambulance Service.** Local Services Welsh Ambulance Service - Frequently Asked Questions. *nhsdirect.wales.nhs.* [Online] [Cited: 2015 04 12.] http://www.nhsdirect.wales.nhs.uk/localservices/welshambulanceservicefaq/#request_an_emergency_.

135. **US Institute of Standards & Technology.** [Online] [Cited: 2014 04 13.] http://www.itl.nist.gov/div897/docs/EHR.htm .

136. **Black, et al., et al.** *The impact of eHealth on the quality and safety of health care: a systematic overview.* s.l. : PLoS Med2011;8:e1000387, 2011.

137. **NHS.** *Intrapartum care: Care of healthy women and their babies during childbirth.* London : National Institute for Health and Clinical Excellence (NHS), 2007.

138. **Oxford University Press.** *Oxford Dictionary of English.* 2005.

139. **SAE ARP 4761.** *Guidelines and Methods for Conducting the Safety Assessment Process on Civil Airborne Systems and Equipment.* s.l. : Society of Automotive Engineers, Inc,Warrendale, PA., 1996.

140. **Habli, I.,, Hawkins, . and Kelly, T.** *Software safety: relating software assurance and software integrity.* s.l. : International Journal of Critical Computer-Based Systems (IJCCBS). 1, 4, (November 2010)., 2010.

141. **Salles, G., et al., et al.** *A Contribution to Organizational and Operational Strategic Alignment:Incorporating Business Level Agreements into Business Process Modeling.* s.l. : IEEE 10th International Conference on Services Computing, 2013 .

142. **Health and Social Care Information Centre, .** *Clinical Risk Management: its Application in the Deployment and Use of Health IT Systems.* s.l. : ISB 0160, 2013.

143. **Bruning, S.,, Weissleder, S. and Malek, M.** *A fault taxonomy for service-oriented architecture.* Dallas, US : High Assurance Systems Engineering Symposium, 2007.

144. **Bruning, S., Weißleder, S. and Malek, M.** *Fault Taxonomy for Service-Oriented Architecture.* Germany : Humboldt-Universit¨ at zu Berlin, , 2007.

145. **Jhumka, A.** *Dependability in Service-Oriented Computing.* London : in Editor (Ed.)∧(Eds.): 'Book Dependability in Service-Oriented Computing' (Springer,edn.), pp. 141-160 , 2010.

146. **Brown, A., Fenn, J. and Menon, C.** Issues and considerations for a modular safety certification approach in a Service-Oriented Architecture. *In 5th IET International System Safety Conference.* 2010 йил, pp. 1 - 6.

147. **Coleman, J, and Jones, C.** *A structural proof of the soundness of rely/guarantee rules.* s.l. : Journal of Logic and Computation, 17, 4, 2007.

148. **Health and Social Care Information Centre, .** *Clinical Risk Management: its Application in the Deployment and Use of Health IT Systems.* s.l. : ISB 0160, 2013.

149. **Seong, P.** *Reliability and Risk Issues in Large Scale Safety-critical Digital Control Systems.* s.l. : Springer Science & Business Media, 2008, 2008.

150. **Arthur, J., et al., et al.** *Fundamentals of Natural Gas Processing.* s.l. : CRC Press,, 2006.

151. **Taylor, J.R.** *Risk Analysis for Process Plant, Pipelines and Transport.* s.l. : Routledge, 2003.

152. **The Offshore Installations (Safety Case) Regulations.** legislation.gov.uk. *legislation.* [Online] 2005. [Cited: 23 12 2014.] www.legislation.gov.uk/uksi/2005/3117/contents/made.

153. **Henderson, Jamie.** Safety case use in the petrochemical industry. [Online] [Cited: 17 12 2014.] http://www.health.org.uk/media_manager/public/75/publications_pdfs/Safety%20cases_supplement%20E.pdf.

154. **Sutton. and Ian.** *Process Risk and Reliability Management: Operational Integrity Management.* s.l. : Gulf Professional Publishing, Edition 2, 2014 .

155. **De Wolf, D. and Mejri, M.** *Crisis communication failures: The BP Case Study.* s.l. : International Journal of Advances in Mangement and Economics, VOL2(Issue.2),48-56., 2013.

156. **kiiitv.** kiiitv. *Southcross Energy Investigating Cause of Natural Gas Plant Fire.* [Online] 2015. [Cited: 22 01 2015.] http://www.kiiitv.com/story/27904949/southcross-energy-investigating-cause-of-natural-gas-plant-fire.

157. **Sutton, Ian.** *Process Risk and Reliability Management: Operational Integrity Management.* s.l. : Elsevier Inc., 2010.

158. **Object Management Group (OMG).** *Service oriented architecture Modeling Language (SoaML) Specification,.* s.l. : Version 1.0.1, May 2012.

159. *Engineering Data Book.* Tulsa, OK, : Gas Processors SupplyAssociation,12th ed., Sec. 2,, 2004.

160. **Peterson, M.J.** *Case study: Bhopal Plant Disaste.* s.l. : Science, Technology & Society Initiative, University of Massachusetts Amherst., 2009.

161. **Slettebø, E,.** *Separation of Gas from Liquids in.* Norway : Master of Science in Product Design and Manufacturing,Norwegian University of Science and Technology Department of Energy and Process Engineering, 2009.

162. **McDermid, J.A.** *Software Safety: Where's The Evidence?* s.l. : Australian Computer Society,, 2001. in Proceedings of the Sixth Australian Workshop on Industrial Experience with Safety Critical Systems and Software, Brisbane, Australia, published in Conferences in Research and Practice in Information Technology Series, P. Lindsay (Ed.), vol. 3, pp.1-6,.

163. **Graydon, Patrick J,, Knigh, John C., and Strunk, Elisabeth A.** *Assurance Based Development of Critical Systems.* s.l. : In: 37th Annual IEEE/IFIP International Conference on Dependable Systems and Networks., 2007.

164. **Hawkins, R.,, et al., et al.** *Weaving an Assurance Case from Design: A Model-Based Approach.* Daytona Beach, Florida, USA, : 16th IEEE International Symposium on High Assurance Systems Engineering,, 2015.

165. **Denney, E.,, Pai, G., and Habli, I.** *Dynamic Safety Cases for Through-life Safety Assurance.* Florence, Italy : 37th International Conference on Software Engineering (ICSE 2015), 2015.

166. **Monakova, G., Brucker, A. and Schaad, A.** *Security and Safety of Assets in Business Processes.* Germany : ACM, 2012.

167. *Enterprise architecture modeling with SoaML using BMM and BPMN - MDA approach in practice.* **Sadovykh, A., et al., et al.** 2010, Software Engineering Conference (CEE-SECR), 2010 6th Central and Eastern European,, pp. pp.79-85, 13-15.

168. **Bruning, S., Weissleder, S. and Malek, M.** *A Fault Taxonomy for Service-Oriented architecture.* Systems Engineering Symposium, pages 367–368, : In HASE '07: Proceedings of the 10th IEEE High Assurance Systems Engineering Symposium, pages 367–368,IEEE Computer Society., 2007.

169. JBoss Community. *jBPM.* [Online] n.d. http://www.jboss.org/jbpm.

170. **Delgado, A, et al., et al.** *A Model-Driven and Service Oriented Framework for the Business Process Improvement.* s.l. : CSSIJournal of Systems Integration, vol. 1, no.3, pp. 45-56., 2010.

171. Activiti Components. *Activiti.* [Online] http://activiti.org/components.html.

172. **NHS, .** *NHS choices.* s.l. : http://healthguides.mapofmedicine.com/choices/map/index.html, 2011.

173. **SAE.** *Guidelines and Methods for Conducting the Safety Assessment Process on Civil Airborne Systems and Equipment.* s.l. : ARP4761, (December 1996)., 1996.

174. **Health and Social Care Information Centre .** *Clinical Risk Management: its Application in the Manufacture of Health IT Systems.* s.l. : ISB 0129., 2013.

175. **Health and Social Care Information Centre, .** *Clinical Risk Management: its Application in the Deployment and Use of Health IT Systems.* s.l. : ISB 0160., 2013.

176. **Fahadullah, M, and Khan, M,.** *Comparison of Hazard Analysis Methods Using a Data Center Example.* Sweden : Department of Computer and Information Science at Linköping University, 2008.

177. **Burns, J., and Pitblado, M.** *A Modified Hazop Methodology For Safety Critical System Assessment.* s.l. : in Proceedings of the First Safety Critical Systems Symposium, Springer-Verlag Symposium, Springer-Verlag , 1993.

178. **Chudleigh, M.** *Hazard Analysis Using Hazop: A Case Study.* s.l. : Proceedings of the 12th International Conference On Computer Safety,Reliability and Security, ed. J. Gorski , 1993.

179. **Earthy, V.** *Hazard and Operability Study As An Approach To Software Safety Assessment.* s.l. : Proceedings of the IEE Colloquium on Hazard Analysis , 1992.

180. **Fenelon, P,, et al., et al.** *Towards integrated safety analysis and design.* s.l. : ACM Applied Computing Review, , 1994 .

181. **McDermid, J,, et al., et al.** *Experience with the application of HAZOP to computer-based systems.* s.l. : In Proc 10th Annual Con} on Computer Assurance, pages 37-48,, 1995.

182. **CISHEC, .** *A Guide To Hazard And Operability Studies.* s.l. : The Chemical Industry Safety and Health Council of the Chemical Industries Association Ltd., 1977.