UNIVERSITY OF TARTU
FACULTY OF MATHEMATICS AND COMPUTER SCIENCE
Institute of Computer Science
Cyber Security

Mohammed AbuLamddi

Developing Secure and Safe Systems with Knowledge Acquisition for Automated Specification

Master's thesis (30 ECTS)

Supervisor: Dr. Raimundas Matulevičius

TARTU 2014

Developing Secure and Safe Systems with Knowledge Acquisition for Automated Specification

Abstract

There are special techniques languages that are used in risk management in both domains of safety engineering and security engineering. The outputs, known as artifacts, of these techniques are separated from each other leading to several difficulties due to the fact that domains are independent and that there is no one unifying domain for the two. The problem is that safety engineers and security engineers work in separated teams from throughout the system development life cycle, which results in incomplete coverage of risks and threats.

The thesis applies a structured approach to integration between security and safety by creating a S*a*S (Safety and Security) domain model. Furthermore, it demonstrates that it is possible to use goal-oriented KAOS (Knowledge Acquisition in automated Specification) language in threat and hazard analysis to cover both safety and security domains making their outputs, or artifacts, well-structured and comprehensive, which results in dependability due to the comprehensiveness of the analysis.

The structured approach can thereby act as an interface for active interactions in risk and hazard management in terms of universal coverage, finding solutions for differences and contradictions which can be overcome by integrating the safety and security domains and using a unified system analysis technique (KAOS) that will result in analysis centrality.

Keywords: Safety information model, security information model, dependability requirements, goal modelling, goal-oriented modelling, KAOS, information systems modelling, Obstacles base.

Turvaliste ja ohutute süsteemide arendamine KAOS meetodi kasutamisel

Lühikokkuvõte

Käesolevas magistritöös rakendatakse struktuurset lähenemist, turvalisuse ja ohutuse integreerimiseks läbi S*a*S (Safety and Security) domeeni mudeli loomise, mis integreerib neid mõlemaid. Lisaks töö käigus näidatakse, et on võimalik kasutada eesmärgipõhist KAOS (Knowledge Acquisition in autOmated Specification) keelt ohtude ja riskide analüüsiks, nii et kaetud saavad nii ohutus- kui ka turvadomeen, muutes nende väljundid e. artefaktid hästi struktureerituks, mille tulemusena toimub põhjalik analüüs ja suureneb usaldatavus.

Võtmesõnad: Ohutusmudel, turvalisusmudel, usaldatavuse nõuded, eesmärgimudel, eesmärgipõhine modelleerimine, KAOS, infosüsteemide modelleerimine, takistus

Acknowledgements

I would like to take this opportunity to express my deep gratitude for my supervisor Dr. Raimundas Matulevičius for his unwavering guidance support and patience throughout his supervision on my thesis. I also would like to thank him for the time he had given me throughout progressing on my thesis keeping his door always open for me for guidance and motivation. This thesis could not have been completed as comprehensively and to the high scholarly degree without his support. I am also grateful for the personal growth and insight gained far beyond the scope of this thesis, and the opportunity to explore new realms of systems thinking.

I would also like to thank Dr. Robert Darimont for giving me a license to use Objectiver tool by Respect-IT, free of charge throughout the time I was working on my thesis.

I would like to thank Col. William Young for his explanations while I was working on the STPA-*sec*.

I am very grateful to my friends from Palestine and Estonia whom I spent fantastic times with, and have motivated me while working on my thesis.

Last but not the least, I would not be able to begin, continue, and complete my graduate school experience without the love and support of my family. My mother, father, brothers, and sisters were influential in achieving this personal goal. Thank you for the steadfast support over the years.

Tallinn-Tartu, 25th May, 2014
Estonia
Mohammad Abulamddi

Table of Contents

List of Figures

List of Tables

List of Abbreviation

AF *Abuse frame*
ALARP *As low As reasonably practicable*
AS/NZS *Australia and New Zealand Standards*
AT *Attack tree*
BDMP *Boolean logic Driven Markov Processes*
CC *Common Criteria*
CIA *Confidentiality, Integrity and Availability*
DiD *Defense in Depth*
FMEA *Failure Mode, Effects and Criticality Analysis*
FTA *Fault Tree Analysis*
GML *Goal Modeling Language*
GQM *Goal, Question, Metric*
HACMS *High-Assurance Cyber Military Systems*
HAZOP *Hazard and Operability*
ICH *International Conference on Harmonisation*
IS *Information System*
ISO/IEC *International Organization for Standardization and the International Electrotechnical Commission*
ISO/TC *International Organization for Standardization and Echnical Committee*
ISRM *Information Safety Risk Management*
ISSRM *Information System Security Risk Management*
KAOS *Knowledge Acquisition in autOmated Specification*
LTL *Linear Temporal Logic*
MDA *Mal-activity diagrams*
MDD *Medical Devices Directive*
MEHARI *Method for Harmonized Analysis of Risk*
MPLS *Multiprotocol Label Switching*
MUC *Mis-use Case*
NIST *National Institute of Standards and Technology*
RE *Requirement Engineering*
SaS *Safety and Security*
SIL *Safety Integrity Level*
STAMP *Systems-Theoretic Accident Model and Processes*
STPA *System-Theoretic Process Analysis for safety*
STPA-Sec *System-Theoretic Process Analysis for Security*
TT *Threat Tree*
UML *Unified Modeling Langu*

CHAPTER 1 Introduction

Our dependability on software in every aspect of our lives has exceeded the level that was expected in the past. We have now reached a point where we are currently stuck with technology, and it made life much easier than before. The rapid increase of technology adoption in the different aspects of life has made technology affordable and has led to an even stronger adoption in the society.

As technology advances, almost every kind of technology is now connected to the network like infrastructure, automobiles, airplanes, chemical factories, power stations, and many other systems that are business and mission critical. Because of our high dependency on technology in most, if not all, aspects of life, a system failure is considered to be very critical and might result in harming the surrounding environment or put human life at risk.

Challenges such as concepts, modelling language and methods used in the fields of safety and security arise during research on either field. The gap between the two field resulted from the fact that researched focuses on either one of those two fields alone, given that each has its own development tools and methods. However, the requirements of safety and security are similar in the fact that they are concerned about what the system-to-be should and should not do.

The scope of the thesis is between safety engineering, security engineering, and risk management for both of them (Figure 1.1). This thesis will address the *information system security risk management* (ISSRM) domain model [Mayer, 2009], as we have contributed in modifying the *information safety risk management* (ISRM) domain model [Firesmith, 2003], the result of integrating the two domains is a safety and security (S*a*S) information domain model.

After that, we will address each domain separately by running the example *@RemoteSurgery* on the security aspect that deals with information security system risk management (ISSRM) [Mayer, 2009] domain, a modified version of information safety risk management (ISRM) [Firesmith, 2003] domain on the safety aspect, and S*a*S that was produced in this thesis.

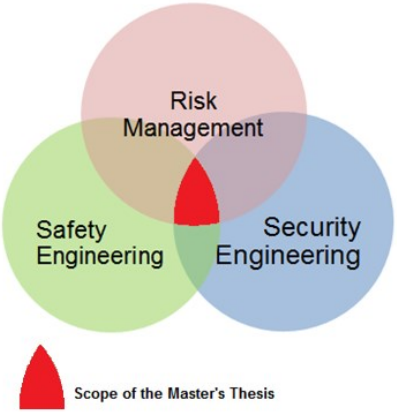

Figure 1. 1 Scope of the thesis.

1.1 Research questions and contribution

In the current thesis, we raise two research questions:

RQ1: *How could we possibly relate safety and security?*

Risk management process was the entry point for the integration process (Figure 1.1) as the interface interplays between safety requirements using information safety risk management domain model (ISRM) and security requirements using information system security risk management domain model (ISSRM) from the aspect of system functionality and what the system should and should not do. For that, we proposed the creation of an information domain model that integrates between safety and security, (S*a*S), and the implementation of risk management process that leads to dependability requirements (safety and security).

RQ2: *What is meant by extending modeling languages approach for safety and security risk management?*

The alignment between S*a*S domain model for hazard management with the modeling language KAOS, which has allowed for a better method to derive safety and security requirements in early stages from the beginning of the system development life cycle. The alignment between S*a*S domain model and KAOS enhances the cooperation and facilitates communication and interaction between stakeholders.

1.2 Running Example

The motivation behind choosing the *@RemoteSurgery* example is that it has focus on both safety and security perspectives, as they affect each other. This example shows what can happen even when the devices are not connected to the network *"In the summer of 2005, radiotherapy machines in Merseyside, England, and in Boston were attacked by computer viruses. It makes little sense to invest effort in ensuring the dependability of a system while ignoring the possibility of security vulnerabilities…."*[Daniel et al., 2007]. In this example, risk on the security side can have a negative effect on safety and can ultimately lead to death.

The example of *@RemoteSurgery* was chosen due to its closeness to real life [Deloitte, 2013], which touches the safety side [Jung and Kazanzides, 2013] because any error in hardware stack console will lead to severe injury or the loss of human life, and the security side because the stack console is connected to the network [Marescaux, et Al., 2002]. These types of risks are also brought out in three following experiments: Operation Lindbergh [Marescaux, et Al., 2002], Operation Canada Tele-Surgeries [Anvari et al., 2005; Anvari, 2007] and remote surgery experiment between Japan-Korea [Jumpei et al., 2006].

Furthermore, *@RemoteSurgery console*s devices run on an operating system like URObot [Fei, et al., 2001] that uses a Linux Red Hat 6.1 distribution using Fast Light Tool Kit (FLTK) in the GUI among other things. We realize that these systems can be infected with viruses and compromised like the rest of the systems, which affects safety.

The researchers [Baowei et al., 2001] have focused on safety in surgeries through evaluation and analysis in terms of the software and hardware used to operate the console. However, they did not take into account the console's connectivity to the network in their evaluation and analysis.

The following description is an extract from Operation Lindbergh [Marescaux, et al., 2002], Operation Canada Tele-Surgeries [Anvari et al., 2005; Anvari, 2007], and experiment between Japan-Korea [Jumpei et al., 2006]. *@RemoteSurgery* consists of three main components; the ***patient information*** that will be shared, the master console that is located in the same operating theatre where the surgeon will be controlling the surgery on his side, and the slave console located next to the patient. The ***slave console*** receives commands from the operating surgeon sent from the ***master console***. These commands are then executed on the patient's body directly without any human interference. The third component is ***telecommunications technology*** used to link the master and slave console in order to transmit the video live feed to the operating surgeon and for the surgeon to send the operating commands to the slave console. Transmitting and receiving operations in this case are subject to packets loss, which puts the operation at risk; furthermore, there is also the risk of packets delay.

Our goal is to address hazards from the safety aspect and threats from the security aspect in a single domain model that integrates the two aspects and performs hazard and threat analysis using KAOS as addressed by the researchers in Operation Lindbergh [Marescaux, et. Al., 2002], Operation Canada Tele-Surgeries [Anvari et al., 2005; Anvari, 2007], experiments between Korea and Japan [Jumpei et al., 2006].

1.3 Structure

This thesis is composed of eight chapters as shown in (Figure 1.2).

Chapters two, three, and five are similar in terms of organization. Each of these chapters is organized as follows: standards and domain models, hazard/risk management process, and finally, techniques languages.

Chapter 2, titled "Safety Engineering", addresses the standards followed by our contribution in adapting information safety risk management (ISRM) domain to support the hazard/risk management process, and finally, safety modelling languages.

Chapter 3, titled "Security Engineering", addresses the information system security risk management (ISSRM) domain, followed by the hazard/risk management process, and finally security modelling languages.

Chapter 4, titled "Knowledge Acquisition in autOmated Specification" in running example '*@RemoteSurgery*'. The example is run on ISSRM and ISRM, the alignment between KAOS and ISSRM. As a part of contribution we did the alignment between ISRM and KAOS.

Chapter 5, titled "Common Method to Define Security and Safety (S*a*S)", is the result of the main contribution in integrating chapters 2 and 3. We addressed the standards and the S*a*S domain produced followed by hazard/risk management process and S*a*S modelling

languages. This chapter also includes an introduction on STAMP approach which is expanded upon in the appendices A, B, and C and a comparison between its results with the results of chapter 6 are discussed in chapter 7.

Chapter 6, titled "Knowledge Acquisition in autOmated Specification for S*a*S" in running example '*@RemoteSurgery'*. The example is run on S*a*S domain, the alignment between KAOS and S*a*S. The results of this process are discussed in chapter 7.

Chapter 7, titled "Validation", consists of the validation and comparison between the uses of KAOS in running example on the suggested S*a*S domain and the use of Systems-Theoretic Accident Model and Processes (STAMP) techniques languages (STPA; Appendix B, STPA-*sec*; Appendix C) running the same example (Chapter 6) on S*a*S domain.

Chapter 8, we provides our conclusions, including limitations of the study and future perspectives.

Appendix A, is an extension of the STAMP Approach section 5.5 in chapter 5. This appendix addresses the Alignment between the concepts of STAMP Approach and S*a*S domain model and detailed explanation on the use of STAMP approach concept using the S*a*S domain model in running the example '*@RemoteSurgery'*.

Appendix B, titled "STPA process for safety" running example '*@RemoteSurgery'* System-Theoretic Process Analysis for safety (STPA) safety corner. This appendix is an extension to chapter 5, the STAMP approach section 5.5, where we use STPA Process for safety in running the example "*@RemoteSurgery*" that has been discussed in chapter 6 by KAOS, in the safety side section. We also use the same description of "running example '*@RemoteSurgery"*" and run it on the safety side using STPA Safety. The results of this process are discussed in chapter 7.

Appendix C, titled "STPA-*sec* process for security" running example '*@RemoteSurgery'* System-Theoretic Process Analysis for security (STPA-*sec*) security corner. This appendix is an extension to chapter 5, the STAMP approach section 5.5, where we use STPA-*sec* Process for security in running the example "*@RemoteSurgery*" that has been discussed in chapter 6 by KAOS, in the security side section. We also use the same description of "running example '*@RemoteSurgery*'" and run it on the security side using STPA-*sec*. The results of this process are discussed in chapter 7.

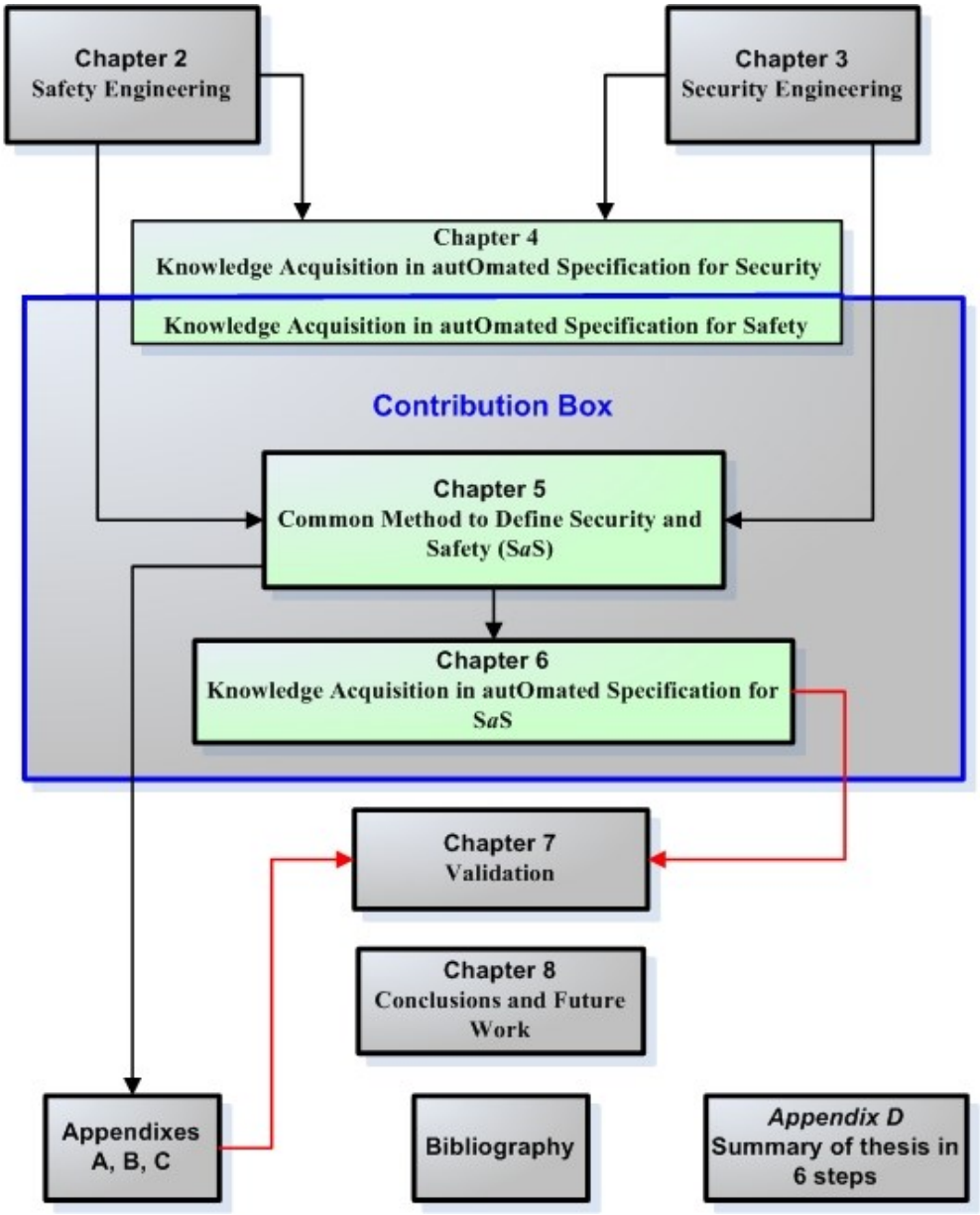

Figure 1. 2 Structure of the thesis.

1.4 Summary of thesis in six steps

Summary of thesis in six steps as shown in (Figure 1.3)

Step 1: KAOS graphical modelling language were used in running the example '*@RunningSurgery'* on the safety side in respect to both the information safety risk management (ISRM) [Firesmith, 2003] domain model and the hazard management process and we did the alignment of the ISRM domain model elements and the KAOS modelling language.

Step 2: KAOS modelling languages were used in running the example '*@RunningSurgery'* on the security side in respect to both the information system security risk management

(ISSRM) [Mayer, 2009] domain model and the risk management process and we did the alignment of the ISSRM domain model elements and the KAOS modelling language.

Step 3: We propose a solution through the creation of a (S*a*S) Safety and Security information domain model that integrates safety and security domains, giving a better opportunity for comparison and integration to find a middle ground between the two domains, as well as unifying definitions through their mappings onto the common concepts.

Step 4: KAOS modelling language were used in running the example '*@RunningSurgery'* on the security and the safety sides in respect to both the S*a*S domain model and the hazard management process and we did the alignment of the S*a*S information domain model elements and the KAOS modelling language.

Step 5: We chose the Systems-Theoretic Accident Model and Processes (STAMP) approach and its modelling language, namely System-Theoretic Process Analysis for safety (STPA), on the safety side and System-Theoretic Process Analysis for Security (STPA-sec) on the security side in order to be the base of the experiment in comparison to what was done in steps 3 and 4.
The concepts of S*a*S domain model were applied on STAMP approach using the same example '*@RemoteSurgery'*.
STPA modelling language were used in running the example '*@RunningSurgery'* on the safety side in respect to both the STAMP domain model and the STPA hazard management process.
STPA-*sec* modelling language were used in running the example '*@RunningSurgery'* on the security side in respect to both the STAMP domain model and the STPA-*sec* hazard management process.

Step 6: We now have the S*a*S domain model and its own modelling language, KAOS-S*a*S, which resulted from the steps 3 and 4. We also have STAMP approach and its modelling language, STPA and STPA-*sec* that resulted from step 5.

Each domain and its own modelling language has been explained along with usage and execution on the same example '*@RemoteSurgery'* followed by the comparison and validation on how and to what extent each domain and its modelling language are covering the safety and the security sides.

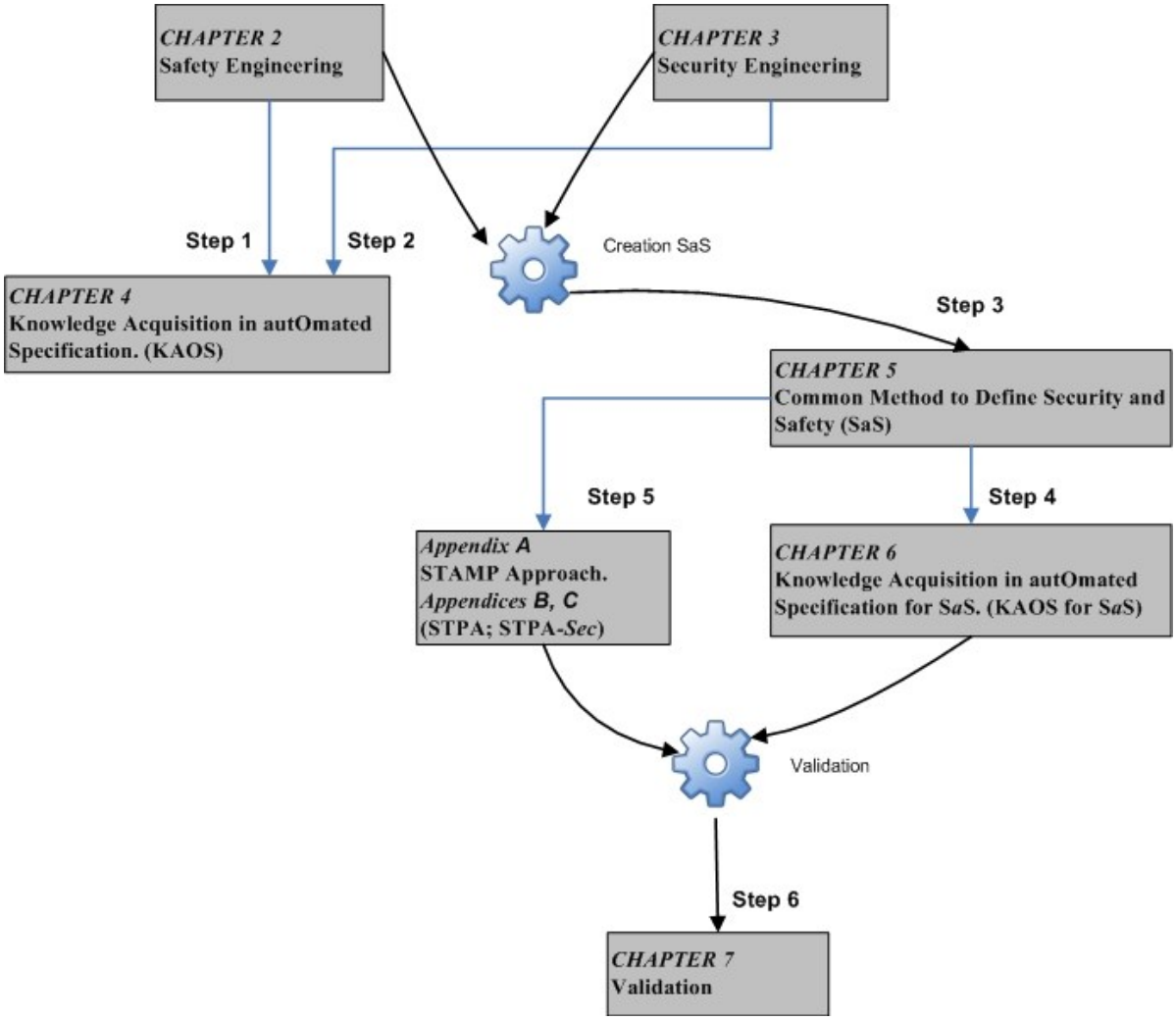

Figure 1. 3 Summary of the thesis in six steps.

CHAPTER 2 Safety Engineering

"The systems engineering discipline concerned with lowering the risk of unintentional (i.e., accidental) unauthorized harm to defended assets to a level that is acceptable to the system's stakeholders by preventing, detecting, and properly reacting to such harm, mishaps (i.e., accidents and safety incidents), system-internal vulnerabilities, system-external unintentional abusers, hazards, and safety risks." [Firesmith, 2012]

We will address the standards used in safety engineering Moreover, we contributed in the adjustment of the domain model of information safety risk management (ISRM) [Firesmith, 2003] model by adding definitions for each artefact and adjusting it to comply with the work being done. We have also addressed safety modelling languages and chose HAZOP and BDMP as each of these two languages has its own techniques for dealing with risk management.

2.1 Software Safety Engineering Standards

It is essential, when implementing critical safety software, that this software is able to verify whether the system is safe or not and it is usually on a high level of verifiability. This is not an easy process as the software systems could be complicated and therefore it would be difficult to determine whether they are truly safe or not. The goals of such standards can be summarized [Hauge, 2001] in the three following points:

Development is the process of putting the new system through the process of defining potential risks and threats in order to discover them and set out a methodology to avoid them.
Operational management it is the process of evaluating risks and threats that have been controlled to reach a higher degree of safety for the system. It is also setting out a clear guide that explains every part of the system and how it to interact with it, and training the users on how to use the system.
Certification is the process of proving that the claimed system has been developed is a safety system and determining the degree of its safety.

DO-178B a standard developed by the Radio Technical Commission for Aeronautics in 1985, the final draft of [DO-178B] and ED-12B was released in 1992. The full name is Software Considerations in Airborne Systems and Equipment Certification.
There are five levels in DO-178B ranging from A to E. These levels describe the consequences of a potential failure: catastrophic, hazardous-severe, major, minor, or no effect.
This certification forces all software requirements to be mapped to a software level describing at which level of criticality the software functions at in possible failure situations. Requirements mapped to a level other than level E are subject to further certification using DO-178B. Requirements mapped with higher levels need very careful planning, coding, and testing. Furthermore, they require more secure configuration

management and to higher levels of quality assurance. The success of producing certified product safety relies on the software levels of DO-178B.

DO-178B divides the software life cycle process into five main processes. Software planning, development, verification, configuration management, and quality assurance. In each level, a number of documents must be produced at before advancing to the next process.

IEC 61508 the IEC 61508 standard [IEC 61508, 1998] developed in 1985 and released in 1999 by The International Electro-technical Commission (IEC) and has the full name 'Functional Safety of programmable electronic systems'.

IEC 61508 is a generic approach involved or used in several industries. Currently, the process industry is developing its own standard that complies with its own specifications for application of Safety Instrumented Systems. IEC 61508 proposes an overall safety lifecycle for software and hardware and addresses all stages. In IEC 61508, safety integrity requirements of the safety functions are specified in four levels in order to allocate them to the Electrical/Electronic/Programmable Electronic (E/E/PE) safety-related systems.

MOD 00-56 standard [MOD 00-56, 1991] was produced by the UK Ministry of Defence in 1991. The full name is 'Defence Standard 00-56: Hazard Analysis and Safety Classification of the Computer and Programmable Electronic System Elements of Defence Equipment'. It describes several forms of hazard and risk analysis to be performed. It is required to carry out a change hazard analysis whenever a change to the system occurs [Hauge, 2001].

This standard gives guidelines for identification, evaluation, and recording the hazards of a system in order for to determine the maximum tolerable risk from it, and to facilitate the achievement of a risk that is as low as reasonably practicable and below the maximum tolerable level. This activity will determine the safety criteria and a reasonable and acceptable balance between reducing risk and the cost of that risk reduction.

MOD 00-56 uses four classes of risk using categories of accident severity and six probability levels to the hazard to determine the class of the risk: intolerable, undesirable, tolerable. However, if the system is being used in a new environment, the hazard classification must be re-performed. There are five approaches to reduce the risk associated with a hazard: re-specification, redesign, incorporation of safety features, incorporation of warning devices, and operating and training procedures [Hauge, 2001].

2.2 Domain Model of Information Safety Risk Management

In the studies conducted by [Firesmith, 2003; Firesmith, 2004; Firesmith, 2005; Firesmith, 2006; Firesmith, 2012] focused on developing the definitions for safety and security domains and comparing them to one another and survivability engineering. They also created a unified definition that includes safety, security, and survivability engineering called defensibility and from that created information models using UML class and founded relationships and definitions between safety engineering and security engineering. However, in the PhD thesis by [Mayer, 2009] included comments on work proposed by

Firesmith that “proposed process does not rely on a risk-based approach” which serves as a motive to create our own domain that works depending on a risk-based approach. (Table 2.2) for Summary of the frameworks.

Firesmith [Firesmith, 2003] distinguishes particularly harm coming from *Intentional* and *Unintentional* source. He then introduces the artifact of defensibility that is defined as the composition of both safety and security, and that is therefore closely related to the scope of our work.

The researchers Axelrod and Mayer commented on Information Safety Risk Management domain model (ISRM) [Firesmith, 2003]. Mayer [Mayer, 2009] said that the ISRM domain does not deal with risk management process while Axelrod [Axelrod, 2012] argued that the concepts of this domain, especially the description of the definitions intentional and unintentional and said that the safety domain should *"Prevent the harmful impact of both accidental and intended hazardous events rather than protect individuals from harm "*.

The reason behind building ISRM domain model is trying to narrow between it and already existing models of security, which will be demonstrated in the security engineering chapter. The safety domain model is easily amenable to hazard analysis and supporting requirement engineering. (Figure 2.1) shows basic definitions on safety engineering like risk, hazard, accident, asset, and vulnerability that have a strong bond with requirement engineering definitions like safety goal, policy and requirement. This explains the public safety and risk analysis methodologies in terms of vulnerabilities, hazards, accidents, and assets. Definitions in (Figure 2.1) are as follows

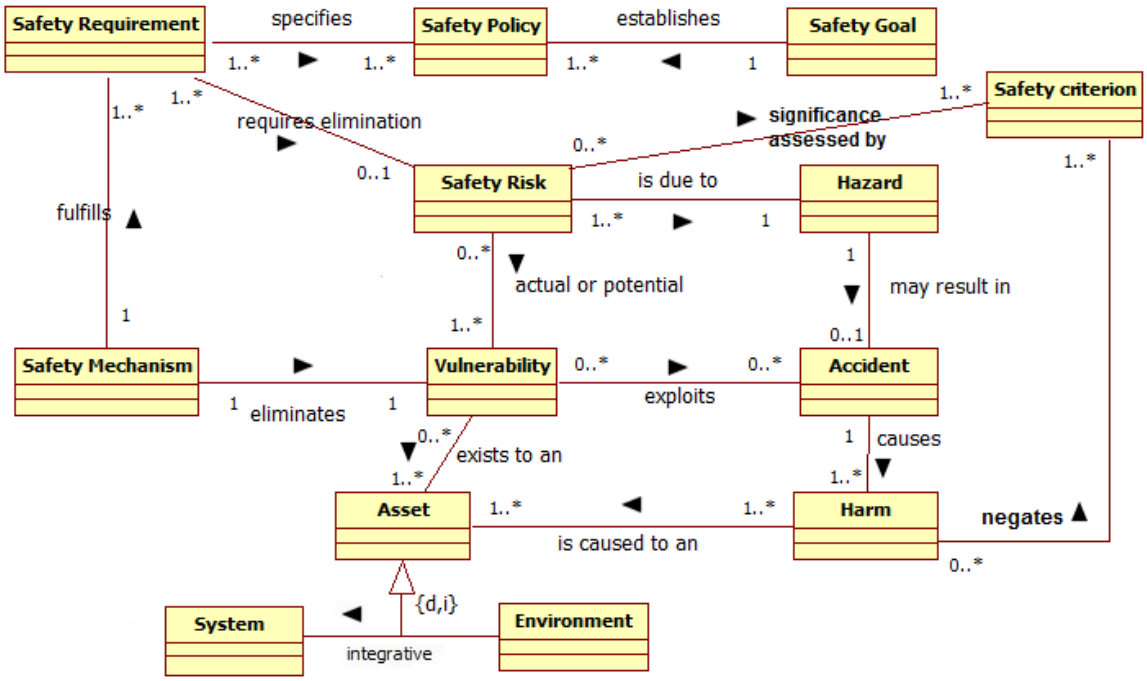

Figure 2. 1 ISRM Domain Model, (Adapted from [Firesmith, 2003]).

Asset the Common Criteria [CC, 2012] define an asset as an “entity that the owner of the target of evaluation places value upon”. Also addresses both [ISO/IEC FDIS 17799, 2005]

and [ISO/IEC 13335-1, 2004] consider "anything that has value to the organization" an asset. And [NIST SP 800-26, 2001] "major application, general support system, high impact program, physical plant, mission critical system, or a logically related group of systems'' an asset.

Therefore we categorised "system" under "Asset" to describe all types of assets of value to the organisation. These systems differ from a company to another whether it's (a.k.a. software, IT infrastructure, users, or strategic plan, etc.).
Taking the surrounding environment into account as in [Zave and Jackson, 1997] where the relationship between the system and the environment "is the portion of the real world relevant to the software development project". And defines the machine "is a computer-based machine that will be constructed and connected to the environment, as a result of the software development project." Furthermore, Jackson and Zave [Zave and Jackson, 1997] go in detail on requirement engineering regarding the environment as when being in an indicated mood the environment is described in absence of the machine and in this case the description comes from a domain knowledge. On the other hand, when in an optative mood, the environment is described as seen fit and as hoped to be achieved when connecting the machine to the environment which is then called requirements in this case.

We prefer to use the definition [ISO/IEC FDIS 17799, 2005] and [ISO/IEC 13335-1, 2004] because it is a broad definition that includes the technical and theoretical aspects like organization reputation and the managerial aspect of organisations. We consider safety to be a system property. Safety can only be regarded as a characteristic of a system. it is not a characteristic of the machine alone.
Valuable asset that may be damaged or destroyed if an accident occurs, e.g. environmental disruptions (accidental disruptions, man-made, natural). Human and operator errors (mistakes by human operators).

Harm is a significant damage, usually associated with an asset that is caused by an accident (when dealing with safety engineering) or is due to an attack (when dealing with security engineering).

Safety risk in the introduction report in [NASA, 1997], Risk is "The combination of the probability (qualitative or quantitative) that a program or project will experience an undesired event and the consequences, impact, or severity of the undesired event were it to occur." And from the safety aspect is quantitative. This representation resulted from the probability of an accident occurring when a system runs in its environment. From this representation accidents are categorised based on the degree of harmness like disastrous or severe for example.

Vulnerability is a weakness in the system that is increases the probability of an accident occurrence that will result in causing harm. This weakness can be in any of the stages of a system development life cycle such as design, implementation, integration, or deployment.

Accident In the introduction report of [NASA 1997] "An unplanned event or series of events that results in death, injury, occupational illness, or damage to or loss of equipment, property, or damage to the environment; a mishap". [IEEE 1228] we can also say that root causes always exist and contribute in the probability of having a sequence of events that would end up with accidents.

Hazard In the introduction report of [NASA 1997] "Existing or potential condition that can result in, or contribute to, a mishap or accident."
The [FAA, 1998] order define hazard as a "condition, event, or circumstance that could lead to or contribute to an unplanned or undesired event."

Hazard Control in the introduction report of [NASA 1997] "Means of reducing the risk of exposure to a hazard. This includes design or operational features used to reduce the likelihood of occurrence of a hazardous effect or the severity of the hazard".

The [ISO 14971, 2012] order define *Hazardous Situation* "circumstance in which people, property, or the environment are exposed to one or more hazard(s)."
Hazardous Situation = Hazard + Sequence of events [ISO 14971, 2012]

Hazard Mitigation in the introduction report of [NASA 1997] "Any action that reduces or eliminates the risk from hazards."

We also notice that the control process in safety is not limited compared to that in security. This is because in safety security, mitigation; control comes from outside the environment which could be resulted from training the employees, or from the rules and regulations.

Safety mechanism are the decisions or plan required to achieve one or more safety requirements and taking them into account throughout the system development life cycle phases, which will decrease the harm caused in case of accidents.

Safety requirement according to the definition in [Zave and Jackson, 1997], A *requirement* is an optative property, intended to express the desires of the customer concerning the software development project. A *specification* is an optative property that specifies a required amount of *Safety Objectives* (Table 2.1) also called *Quality subfactor* [Firesmith, 2012; Romani et al., 2009], intended to be directly implementable and to support satisfaction of the requirements.

Table 2. 1 Concepts Safety Criteria (Safety Quality subfactor) . (Adapted from [Firesmith, 2012; Romani et al., 2009]).

Safety
Fail-safe
Failure tolerance
Survivability
Performance
Robustness
Correctness
Accuracy
Traceability
Recoverability
Human backup

Safety policy multiple requirements are interdependent and interact with one another. These interactions may be positive, negative. The safety policy states, In the event of

conflict between security requirement and safety requirement it shall always be presumed that safety has precedence.

Safety Goal it is the dire need to achieve the highest level of safety in a system as possible. This need comes from the strong motivation behind creating a safety policy if the goal was to achieve high levels of safety in a system where safety policy always gives priority to safety requirements in case of requirements conflicts.

Table 2. 2 Summary of the frameworks and standards safety engineering.

Reference	Safety Oriented	Risk-Based Approach	RE Approach
DO-178B	++	++	--
Firesmith	-+	-+	++
EC 61508	++	++	--
MOD 00-56	++	++	--

Legend:
++: Completely covered and at the core of the document
+- : Partially covered or not playing a central role
--: Not covered

2.3 ISRM Hazard Risk Management Process

Information safety domain model put by [Firesmith, 2003] that addresses safety engineering and the creation of a conceptualised domain model specific for safety and discussed its concepts. He had also done the same for security integrated them into what he called survivability engineering. These domains are built similarly to the system development life cycle as it mainly depends on regular activities of requirement engineering for both safety engineering and security engineering. However, the steps or the risk management processes produced by Firesmith are not clear in the information models.

We elicited these six steps process (Figure 2.2) for risk management from the safety perspective through [Axlerod, 2012; Redmill, 1999] standard IEC 61508 and [Brazendale, 1995] IEC 1508 standard that explain the phases of the hazard risk management process from the safety perspective taking into account the respect to the safety information models [Firesmith, 2003]. The following steps are (a to f) summarised are follows:

(a) Scope and asset identification the first step consists of the process of searching for stakeholders to address the safety implications, at the system level and their environments (a.k.a. physical, social, standards) for the purpose of defining the scope. After that, the assets of value for the company as well as the assets related to safety engineering need to be identified. The output of this step is the definition of the scope and its relation to the system and the environment and a priority list and rankings of assets to be secured from a safety perspective starting with the assets of the highest priority.

(b) Determination of quality factor objective in this step, we set a quality criterion for every asset identified in the previous step, while each asset has its own characteristics, which requires the identification of safety goals for each of these assets as summarized in (Table 2.2).

(c) Hazard analysis and assessment the third step consists of the identification of existing and potential hazards that are likely to violate the safety goals resulting in accidents. Without doubt, these accidents will cause damage to assets. After identification, these hazards are evaluated and the degree of risk is measured using quantitative and qualitative analysis. At this stage, the defining the likelihood of occurrence, defining consequence categories, and risk matrix are produced and the result is full information on these hazards. After that, ALARP principle is implemented to measure the tolerance of each hazard [Redmill, 1999]. If the results are dissatisfying, the entire process has to be performed again starting from step (a), otherwise, the process proceeds to step (d).

(d) Hazard treatment in this step the decision is made regarding these hazards. These types of risk treatments are divided to three categories: prevention, reducing, or retaining risk.

(e) Quality requirements definition depending on the decision(s) made and choosing the measures in the previous step, we derive the safety mechanism, and the strategic decision that will satisfy safety requirement to define Safety Integrity Level (SIL) target that complies to what has been chosen in order to mitigate and control harms resulting from hazards.

(f) Constraint selection and implementation in this step, the decisions made regarding hazards are implemented by setting constraints that comply to SIL target in parallel with implementing safeguards for unintentional hazards. To ensure the compatibility of the chosen quality criterion for each asset individually by referring to the safety policy.

Safety systems are dynamic and interactive resulting in having unintentional hazards. The upgrading process is continuous as the main objective of this step is to monitor the residual risk and its compliance to the standards [Brazendale, 1995].

2.4 Safety Modelling Languages

Most of the techniques mentioned in this research were built specifically for a certain industry, for example Hazard and Operability (HAZOP) that was built and used in chemical industry [IEC61882, 2002], Fault Tree Analysis that was built and used in nuclear industry [Vesely et al., 1981], and The Failure Mode and Effect Analysis that was used in rocket and automobile industries. It is important to note that each and every technique built and used in a specific industry has its own threat analysis and mathematical formulas even if they all under the safety engineering umbrella.

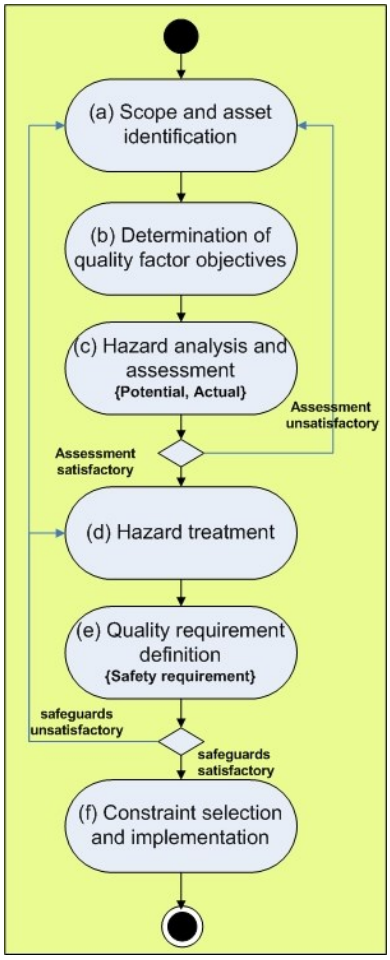

Figure 2. 2 Hazard Risk Management Process, (Adapted from [Axelrod, 2012; Mayer, 2009]).

Hazard and Operability (HAZOP) technique is used in identifying and analysing the threats and risks that can arise during the developed system's operations. This technique is flexible mainly because of the use of guidewords that are adjusted depending on the industrial environment it will be working in. This has resulted in the spread of using guidewords brainstorming process in industries other than the chemical industry. (Table 2.3) shows guideword interpretations for attributes of Messages.
It is possible to apply HAZOP during an early stage of system construction where the main features and behaviour of a system but no details or modules have been produced.

Table 2. 3 Example of Suggested guideword interpretations for attributes of Messages. (Case study from [Klaus et al., 2004]).

Entity=Message		
Attribute	**Guide word**	**Interpretation**
predecessor/ successor	*No*	Message is not sent when it should be.
	Other than	Message sent at wrong time.
	As well as	Message sent at correct time and also at incorrect time.
	Sooner	Message sent earlier within message sequence than intended.
	Later	Message sent later within message sequence than intended.
sender/ receiver	*No*	Message not sent when intended (to any destination).
	Other than	Message sent to wrong object.
	As well as	Message sent to correct object and also an incorrect object.
	Reverse	Source and destination objects are reversed.
	More	Message sent to more objects than intended.
	Less	Message sent to fewer objects than intended.

HAZOP studies The recommended steps in a HAZOP study, which is based on examining design representations of a system, are: identifying each entity in the design representation; descripting the interaction between the components of a component affecting its operation like flow of data for example; applying guidewords to attributes by investigating deviations from the design; investigating the causes and consequences of each deviation; and descripting the recommended mitigations for the hazard.

Boolean logic Driven Markov Processes (BDMP) technique Several researches [Cambacédès and Bouissou, 2010] focused on finding new methods to deal with modeling safety and security interdependencies with BDMP, a technique that depends on graphical modeling and mathematical formalism (Figure 2.3). However, using this newly founded method is impractical because it requires knowledge and hands-on experience because it is very much similar to attack tree and fault-tree with Markov processes.

The ability to formulate BDMP enables modelling dynamic feature with triggers. BDMP is used to model the different combinations of events that may lead to undesired events, which can be system failure for example. In a tree, these events represent the leaves. Each leaf is associated to a "triggered Markov process" that models its different states. This process can

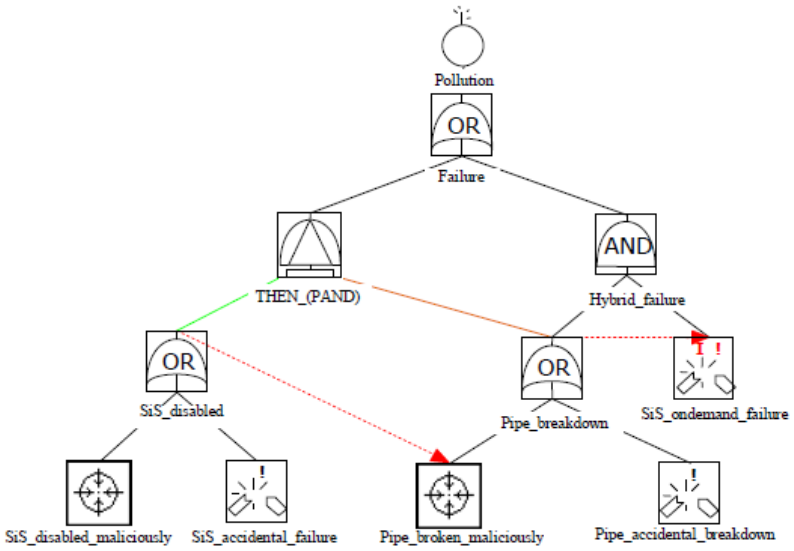

Figure 2. 3 Modeling safety and security interdependencies with BDMP (from [Cambacédès and Bouissou, 2010]).

be in a *Required* and *Not-Required* mode or in an *Idle* or *Active* mode for safety-related and security-related leaves respectively. This method, besides other outputs, gives quantitative results including the sequences that most probable lead to unwanted events.

BDMP is suitable for risk evaluation process and it consists of three phases:

1. *Context definition* we define the scope and boundaries of a system and the nature of the risks will be examined.
2. *System description addressing risks* documenting the scheme of the system intended to be built and its functions.
3. *Risk estimation* this phase consists of three sub-phases: analysing data, representing and modelling system related risks, and exploiting the model.

Choice of prevention and mitigation: this phase depends on quantitative and qualitative risk estimation.

The newly founded technique was derived from a real case study used in [Kriaa et al., 2012] where the focus was on modeling case study about transporting a polluting substance with BDMP hoping towards more formal risk assessments.

We will use KAOS language in chapter 4 section 4.3 to represent and run the example *@RemoteSurgery* in the safety side.

2.5 Summary

In this chapter, we have addressed the safety standards followed by our contribution in adapting information safety risk management (ISRM) domain to support the hazard management process, and finally, safety modelling languages. And the ISSRM domain and hazard management process will be used in running the example *@RemoteSurgery* using KAOS modelling languages from the safety side.

CHAPTER 3 Security Engineering

"The systems engineering discipline concerned with lowering the risk of intentional (i.e., malicious) unauthorized harm to defended assets to a level that is acceptable to the system's stakeholders by preventing, detecting, and properly reacting to such harm, civilian misuses (i.e., attacks and security incidents), system-internal vulnerabilities, system-external intentional civilian abusers, threats, and security risks." [Firesmith, 2012]

Security engineering also includes the organizations goals, strategies, tools, policies, rules, regulations, methodologies and operations that are taken into consideration throughout the system development process to protect it from threats that might occur both from internal and external environments [Bishop, 2004]. The core of security engineering can be summarised in confidentiality, integrity and availability (CIA).

3.1 Domain Model of Information System Security Risk Management

Information System Security Risk Management (ISSRM) is a methodology that focuses on issues related to information systems security risk management. The model is defined after surveying risk management, the security related standards, risk management methods, and software engineering [Mayer, 2009; Mayer et al., 2007]. The domain model shown in (Figure 3.1) supports security modelling languages alignment that also improves security and modelling languages because it is compatible with security threat management for organisations.

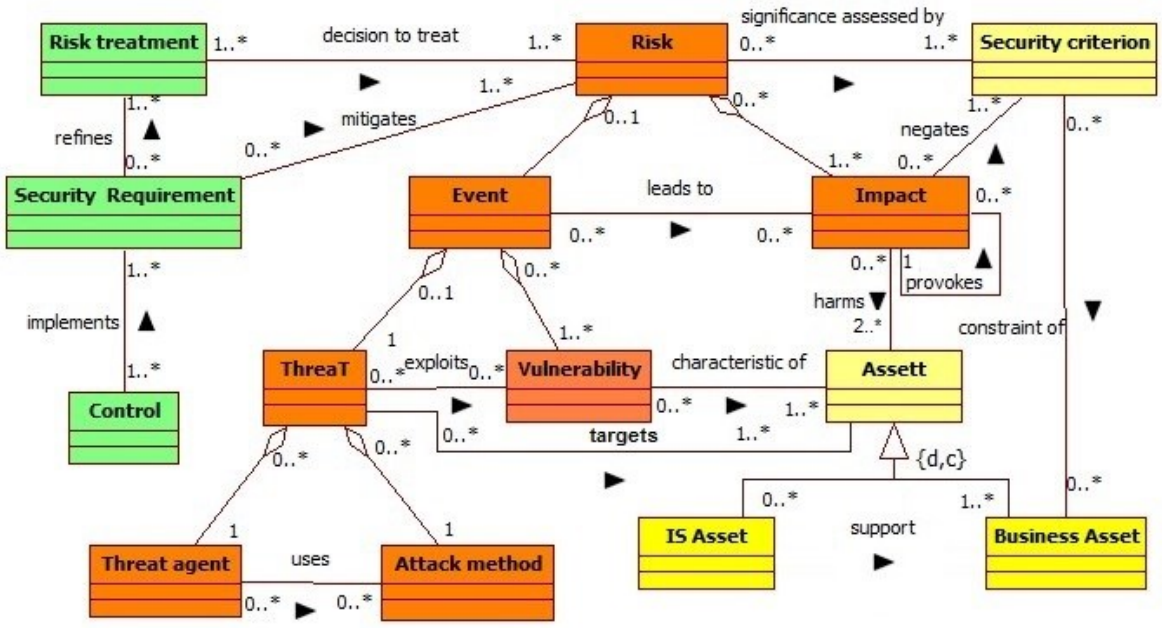

Figure 3. 1 ISSRM Domain Model, (Adapted from [Mayer, 2009]).

The reason why we chose ISSRM model is because of what Mayer [Mayer, 2009] has shown us in the ISSRM domain model covering and intersecting with other security risks management domains, which can be adapted to work with standards and domains such as CORAS [Vraalsen et al., 2007], CRAMM [Insight Consulting, 2003], OCTAVE [Alberts and Dorofee, 2001], MEHARI [CLUSIF, 2007], and NIST 800-30 [NIST SP 800-30, 2002].

Here, the principles and definitions that have been extracted and adapted by Mayer [Mayer, 2009] from the following standards ISO/IEC, AS/NZS, ISRM [Firesmith, 2003], NIST 800-30 and will be commented on. It is important to mention that the standard AS/NZS is adopted and used in ISO/IEC Guide 73 definitions so there is no real addition to AS/NZS standard. Definitions in (Figure 3.1) are as follows according to ISSRM [Mayer,2009].

Asset *"anything that has value to the organisation and is necessary for achieving its objectives. Examples: technical plans; project management process; architectural competences; operating system; Ethernet network; people encoding data; system administrator; air conditioning of server room.*
NOTE: This concept is the generalisation of the business asset and IS asset concepts."

Business asset *"information, process, skill inherent to the business of the organisation, that has value to the organisation in terms of its business model and is necessary for achieving its objectives.*
Examples: technical plans; structure calculation process; architectural competences."

IS asset *"a component or part of the IS that has value to the organisation and is necessary for achieving its objectives and supporting business assets. An IS asset can be a component of the IT system, like hardware, software or network, but also people or facilities playing a role in the IS and therefore in its security.*
Examples: operating system; Ethernet network; people encoding data; system administrator; air conditioning of server room."

Since we are speaking from the security perspective, we need to categories each asset depending on its needs according to the confidentiality, integrity and availability levels. This will help us determine the risks each asset will face, separately.

Security criterion (also called security property; security need) *"property or constraint on business assets characterising their security needs. Security criteria act as an indicator to assess the significance of risk. Security criteria are most often confidentiality, integrity and availability, but sometimes, depending on the context, some other specific criteria might be added, like authenticity, non-repudiation or accountability."*

Risk *"the combination of a threat with one or more vulnerabilities leading to a negative impact harming one or more of the assets. Threat and vulnerabilities are part of the risk event and impact is the consequence of the risk.*
Examples: a cracker using social engineering on a member of the company, because of weak awareness of the staff, leading to non-authorised access on personal computers and loss of integrity of the structure calculation process; a thief penetrating the company's

building because of lack of physical access control, stealing documents containing sensitive information and thereby provoking loss of confidentiality of technical plans."

We agree with AS/NZS definition, as it is more comprehensive on the fact that it is possible for a risk to be either positive or negative. For example, enterprises usually take the risk but this risk is under control and is a positive risk. Furthermore, the ISO/TC 262 definition mentions that the negative risk is closer and more suitable for safety engineering than security engineering.

Impact "*the potential negative consequence of a risk that may harm assets of a system or an organisation, when a threat (or an event) is accomplished. The impact can be described at the level of IS asset (data destruction, failure of a component, etc.) or at the level of business assets, where it negates security criteria, like for example: loss of confidentiality of an information, loss of integrity of a process, etc.*
Examples: password discovery (IS level); loss of confidentiality of technical plans (business Level)."

Event "*the combination of a threat and one or more vulnerabilities.*
Examples: a cracker using social engineering on a member of the company, because of weak awareness of the staff; a thief penetrating the company's building because of lack of physical access control."

Vulnerability *"characteristic of an Information System (IS) asset or group of IS assets that can constitute a weakness or a flaw in terms of IS security. It could be accidentally or intentionally exploited by a threat.*
Examples: weak awareness of the staff; lack of physical access control; lack of fire detection."

The available definitions are considered to be clear definitions of vulnerability. However, in NIST SP 800-30, the definition explains very precisely what we are looking for; the addition of the word *intentionally* and also agreeing with Firesmith's [Firesmith, 2003] definition. The word *intentionally* is what differentiates between safety engineering and security engineering.

Threat "*potential attack or incident, carried out by an agent that targets one or more IS assets and that may lead to harm to assets. A threat is usually composed of a threat agent and an attack method.*
Examples: a cracker using social engineering on a member of the company; a thief penetrating the company's building and stealing media or document."

Firesmith addresses that the likelihood of a threat occurring while Common Criteria (CC's) definition is more comprehensive as it also includes threat agents.

Threat agent "*an agent that can potentially cause harm to assets of the IS. A threat agent triggers a threat and is thus the source of a risk.*
Examples: member of the personnel with little technical ability and time but possibly a strong motivation to carry out an attack; cracker with considerable technical ability, well-equipped and strongly motivated by the money he could make.
NOTE: It can be characterised by its type (usually human or natural/environmental) and by the way in which it acts (accidental or deliberate). In the case of an accidental cause, it can

also be characterised by exposure and available resources and in the case of a deliberate cause, it can also be characterised by expertise, available resources and motivation."

Attack method "*standard means by which a threat agent carries out a threat.*
Examples: system intrusion; theft of media or documents."

Risk treatment *"the decision of how to treat identified risks. A treatment satisfies a security need, expressed in generic and functional terms, and can lead to security requirements. Categories of risk treatment decisions include:*
Avoiding risk (risk avoidance decision) decision not to become involved in, or to withdraw from, a risk. Functionalities of the IS are modified or discarded for avoiding the risk;
Reducing risk (risk reduction decision) action to lessen the probability, negative consequences, or both, associated with a risk. Security requirements are selected for reducing the risk;
Transferring risk (risk transfer decision) sharing with another party the burden of loss from a risk. A third party is thus related to the (or part of the) IS, ensuing sometimes some additional security requirements about third parties;
Retaining risk (risk retention decision) accepting the burden of loss from a risk. No design decision is necessary in this case.
Examples: do not connect the IS to the Internet (risk avoidance); take measures to avoid network intrusions (risk reduction); take an insurance for covering the loss of service (risk transfer); accept that the service could be unavailable for 1 hour (risk retention).
NOTE: Risk treatment is basically a shortcut for risk treatment decision."

Security requirement a*" condition over the phenomena of the environment that we wish to make true by installing the IS, in order to mitigate risks.*
Examples: appropriate authentication methods shall be used to control access by remote users; system documentation shall be protected against unauthorised access."

It should be noted that it is difficult to answer the question of security requirements with yes or no because until now, security requirements are dealt with as whether they are non-functional and the quality factor. For that, to get the best results, security requirements should be dealt with clarity and declare them in the beginning of the requirements gathering phase.

Control (also called countermeasure or safeguard) *"a designed means to improve security, specified by a security requirement, and implemented to comply with it. Security controls can be processes, policies, devices, practices or other actions or components of the IS and its organisation that act to reduce risk.*
Examples: firewall; backup procedure; building guard."

Now that the domain of security engineering is covered, it is possible for us to jump to techniques and security methodologies in which definitions will be treated as introduced in ISSRM.

3.2 ISSRM Risks Management Process

The ISSRM domain model is responsible for the risk assessment management process through three main concepts discussed each separately by Mayer [Mayer, 2009] and they are as follows: (i) asset-related concepts; (ii) risk-related concepts; and (iii) risk treatment concepts. Using these three concepts, Mayer [Mayer, 2009] has put six steps (see Figure 3.2) for the risk management process for the security requirement engineering. The following steps are (a to f) summarised as follows.

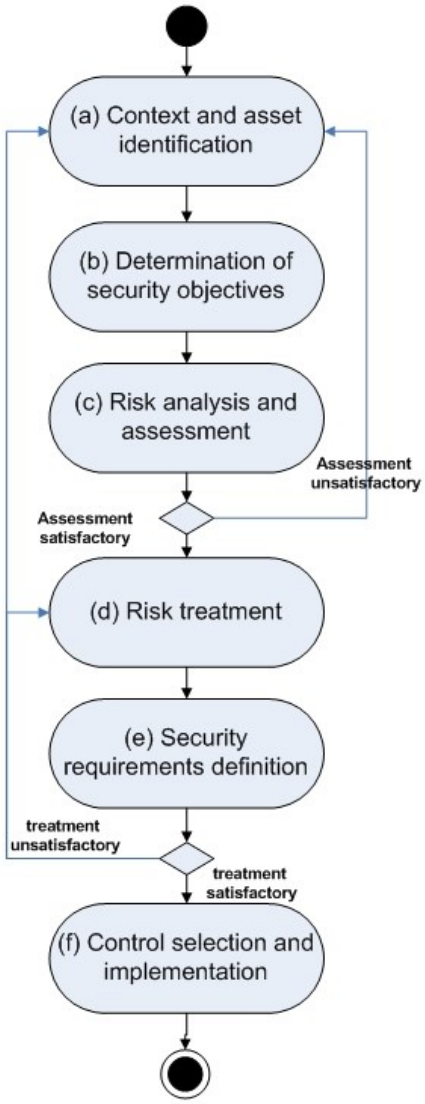

Figure 3. 2 ISSRM Process, (Adapted from [Mayer, 2009]).

(a) Context and asset identification the first process in this step is the search by multiple specialised teams for what is considered valuable for the company such as business assets and IS assets and what the processes the company wants to protect are. Ideally, a priority list of the assets that need security protection where said assets are arranged from the most important and are assigned the highest priority to the least important for the company.

(b) Determination of security objectives in this step, we set up a criterion for every asset identified in the previous step such that every asset has unique requirements, which requires security goals for every asset to be identified and are usually CIA.

(c) Risk analysis and assessment the third step is all about identifying the existing and potential risks that will violate any of the security goals, which will result in damaging the assets. After that, the degree of this risk is evaluated and measured by quantitative and qualitative analysis. The measurement and evaluation stop when the results are satisfying.

(d) Risk treatment decisions regarding risks that have been measured and evaluated in the previous step are made in this step. There are four types of risk treatment: avoiding, reducing, transferring, or retaining risk [Mayer, 2009].

(e) Security requirements definition depending on the decision(s) made in the previous step and choosing the risk treatment type, the identification and derivation of the security requirements that work with the has been chosen to mitigate threats resulting from risks.

(f) Control selection and implementation this is the last step of the process, in which, the implementation of the decisions made regarding mitigating and controlling risks and enhancing the information security level in the company through implementing countermeasures.

3.3 Security Modelling Languages

There are several kinds of technology used in the safety domain, in the analysis of the system development life cycle in general and specifically in analysing potential risks that will obstruct the system to be developed. If we take misuse-case from UML-based approaches, and KOAS from Goal-oriented approaches. We would like to point out an Alignment of misuse cases with ISSRM domain model has been built and the details can be found in [Matulevičius et al., 2008], and also Alignment of ISSRM domain model and KAOS. The detail of the concept alignment between KAOS and ISSRM domain model can be found in [Mayer, 2009].

Misuse case diagrams, the conception of use cases is used to create and relate corresponding misuse cases used to address particularly security requirements [Sindre and Opdahl, 2005]. The functionality of a system is modeled in use cases focusing on interactions with users and responses from the system. Misuse cases extend the positive use cases with the negative ones to ensure eliciting security requirements.

A use case and a misuse case are related using a directed association (Figure 3.3). If the association points from a misuse case to a use case has the stereotype <<*threaten*>> while if the association points from a security use case to a misuse case has the stereotype <<*mitigate*>>. It is stated that ordinary use cases represent requirements, security cases represent security requirements, and misuse cases represent security threats. The essence of the contained use cases is captured in an associated textual description since use case diagrams only give an overview of the system functionality.

Misuse cases are applicable to design a system that covers different security needs. It is possible to consider all three CIA goals. It incorporates common risk and threat analysis techniques.

The process consists of five steps [Sindre and Opdahl, 2005], which consists of (1) Identify critical assets in the system, (2) Define security goals for each asset, (3) Identify threats for each security goal, (4) Identify and analyze risks for the threats, (5) Define security requirements using mitigate.

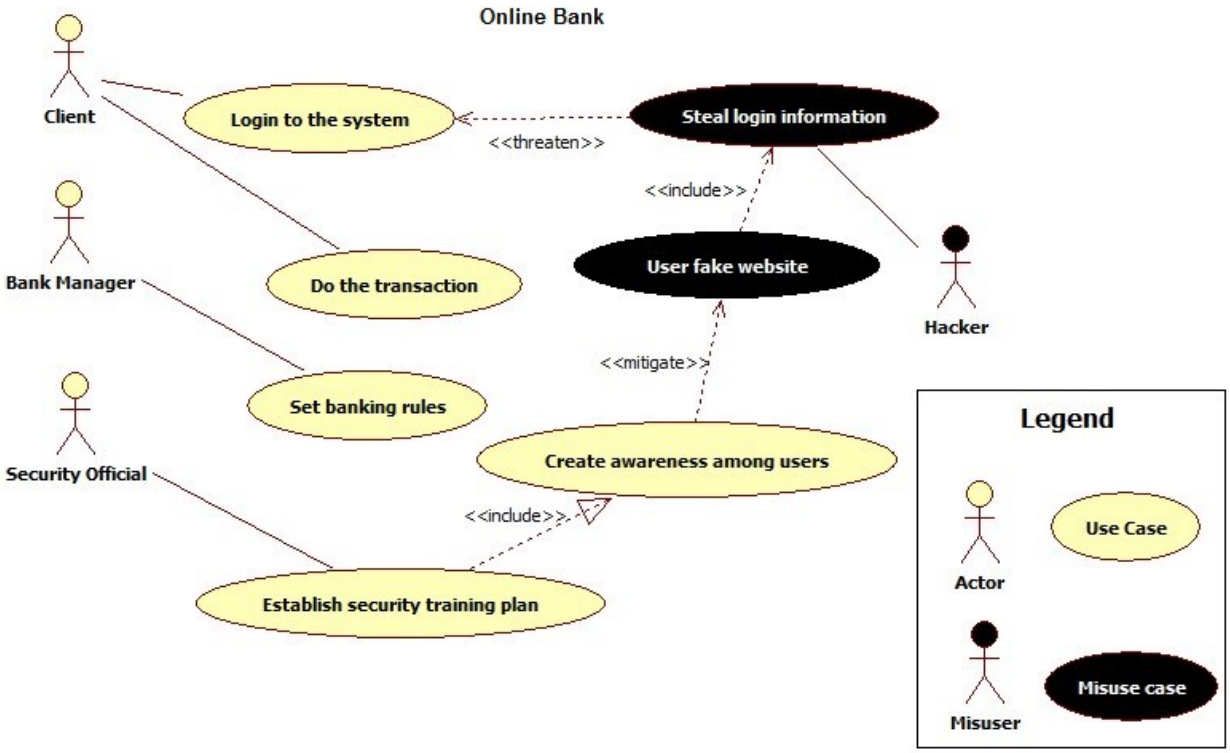

Figure 3. 3 Misuse Case Diagrams login online banking system, (adapted from[Chowdhury, 2011]).

In both studies conducted by [Sindre and Opdahl, 2001; 2005] it is apparent that it is possible to derive a used technique from an existing one to cover the lack in that technique as they have managed to derive misuse case technique from the use-case technique and used it in security engineering.

[Stålhane and Sindre, 2008] have prepared for an experimental comparison between use case diagrams and textual use cases in defining safety hazard identification. Their experiment concludes that using the textual use cases they were able to identify more failure modes or threats than use case diagrams.

[Stålhane, et al., 2010] have conducted two separate experiments to compare between sequence diagrams and textual use cases in hazard identification to find out which is more appropriate in discovering risks that might appear during the early stages of the system

development life cycle and concluded that sequence diagrams are better for the identification of hazards than textual use cases.

Mal-activity diagrams [Sindre, 2007] are based on misuse cases, malicious activities while actors are added to the diagrams to model potential attacks.

It deals with behavioural features of the security problems. A basic way to build a mal-activity diagram is to build a normal process and add the undesired behaviour against this process. This allows the addition of extra concepts (see Figure 3) such as *Mal-Activity*, *Mal-swimlane* and *Mal-decision* and defines *MitigatingActivity* and *MitigatingLink* to show the mitigation process.

The process consists of four steps [Chowdhury et al., 2012], (see Figure 3.4) which consists of (1) Asset Identification, (2) Risk Analysis, *Mal-swimlane* "*hacker*" and malicious actor, (3) Identify threats for each security goal, (4) Define security requirements using Mitigation Activity *Security module.*

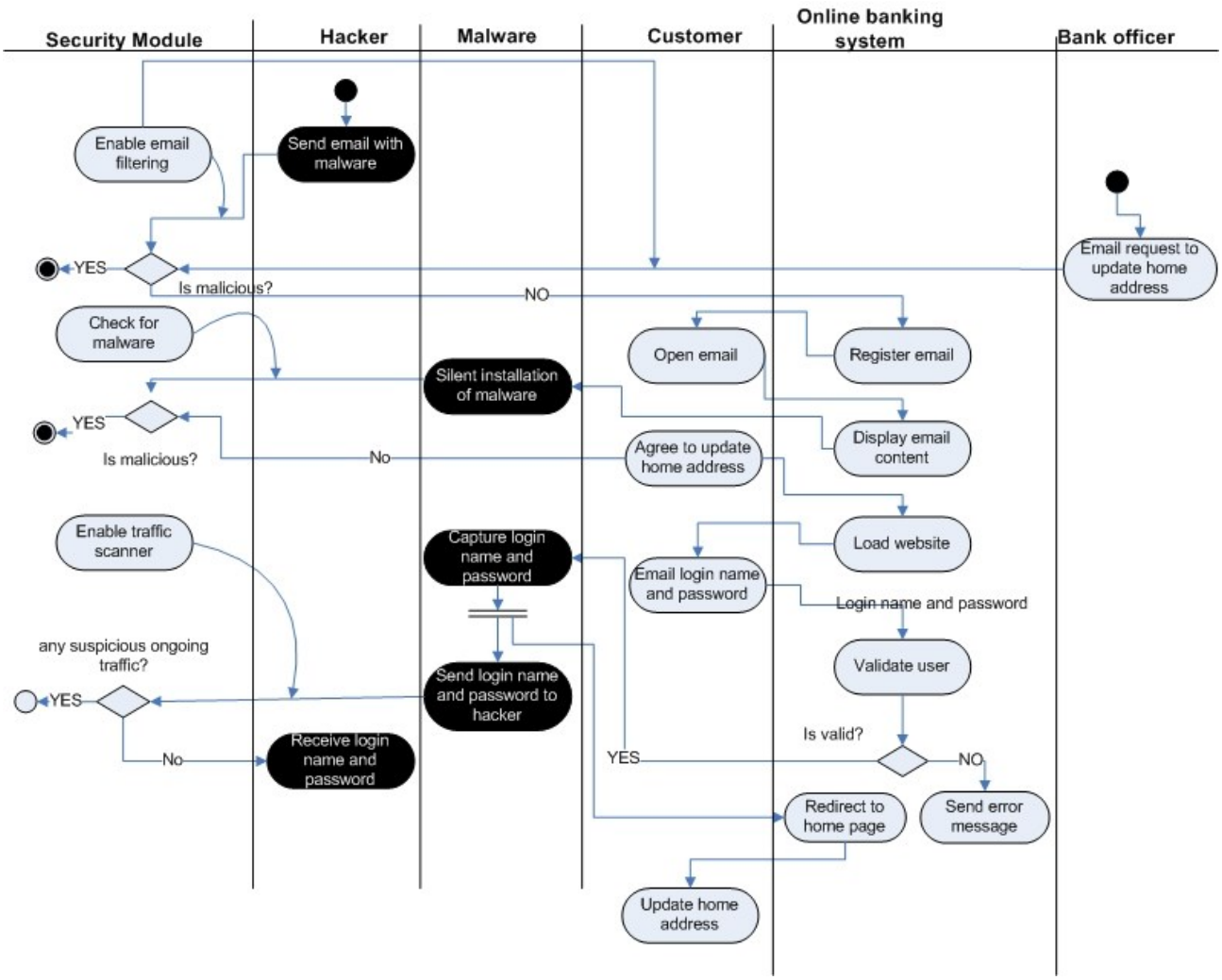

Figure 3. 4 Security Requirements Definition -risk treatment decision- example Online banking system ,(Adapted from [Chowdhury et al., 2012]).

Mal-activity diagrams (MDA) is the best graphical system for modelling misuse-cases because it gives an overview on in-depth risk analysis through which events can be traced from the beginning to the end. Misuse-cases are better than MDA in using textual description and are also easier to use than MDA.

We will use KAOS language in chapter 4 section 4.2 to represent and run the example *@RemoteSurgery* in the security side.

3.4 Summary

In this chapter, we have addressed the information system security risk management (ISSRM) domain, followed by the risk management process, and finally security modelling languages. And the ISSRM domain and risk management process will be used in running the example *@RemoteSurgery* using KAOS modelling languages from the security side.

CHAPTER 4 Knowledge Acquisition in autOmated Specification

KAOS were used in running the example '*@RunningSurgery'* on the safety side in respect to both the information safety risk management (ISRM) [Firesmith, 2003] domain model and the hazard management process and we did the alignment of the ISRM domain model elements and the KAOS modelling language.

KAOS were used in running the example '*@RunningSurgery'* on the security side in respect to both the information system security risk management (ISSRM) [Mayer, 2009] domain model and the risk management process and we did the alignment of the ISSRM domain model elements and the KAOS modelling language.

4.1 Graphical modelling Language

Knowledge Acquisition in autOmated Specification (KAOS) is a methodology for requirements engineering that enables analysts to build requirements models and to derive requirement documents from KAOS models. The meta-model for KAOS has been discussed in [Matulevičius et al., 2006]. KAOS is a goal-oriented requirements engineering method intend to support the entire process of requirements analysis and elaboration – from high-level goals that need to be achieved to the requirements, objects and operations notions assigned to various agents notion in the composite system. It also provides a specification language, a tool support, and an elaboration method [Lamsweerde and Letier, 2000].

The *Goal* model of KAOS looks like a tree that expresses relationships among goals of a system by showing how low-level goals contribute to higher-level goals and how, in this goal model, an AND-refinement link relates a parent goal to a set of sub-goals that must be satisfied for the parent goal for be satisfied. Using KAOS goal refinement patterns are considered an efficient way to build the model because proofs can be reused. These patterns are capable of reducing time and cost of goal model construction.

An *Obstacle* is like a goal. However, the two are used to represent safety goals to reach obstacle treatment through the refinement into sub-obstacles. Each of these sub-obstacles is anchored with a new goal that works towards limiting and treating these obstacles. This method is implemented on the rest of the sub-obstacles until the goal "obstacle treatment" is achieved, which is the main goal and is located on the top level of the KAOS diagram.

The following are definitions of elements found in KAOS (Figure 4.1)

1. *Goals* descriptive milestones statements intended to be achieved.
2. *Agents* active components like humans, devices, and legacy software that play a role towards achieving goals, *Student; UniversityOfficeOfAdmission; CouncilScholarships*

3. *Obstacle* a condition if satisfied, may prevent a goal from being achieved and is used in producing an anti-model that shows why and by whom the original model can be threatened; *[Students NOT know about it Resolution]*.
4. *Requirements* a terminal goal that an agent is responsible for in the software to be developed, *Maintain[Students]*.
5. *Object* any entity defined in the system. An object has features and relations, *Students; CouncilScholarships; UniversityOfficeOfAdmission.*
6. *Action* the interaction between inputs and outputs within an object. Each action has pre-action, post-action, and trigger conditions, *registeredAt; Grants; Partner.*
7. *Operation model* description of all behaviours whose requirements need to be fulfilled by agents. Behaviours are expressed in terms of operations that agents performed. Operations work on objects: they can create objects, trigger, state transitions of objects, and activate other operations [Respect-IT, 2007], *Student; CouncilScholarships*; *UniversityOfficeOfAdmission.*
8. *Responsibility model* the responsibility model contains all responsibility diagrams. Each diagram describes the requirements and expectations an agent is responsible for, or has been assigned to them. An agent is assigned to expectations in a goal model [Respect-IT, 2007], *UniversityOfficeOfAdmission.*

The following (Figure 4.1) addresses the main components of KAOS, which will be reflected upon getting to know them.

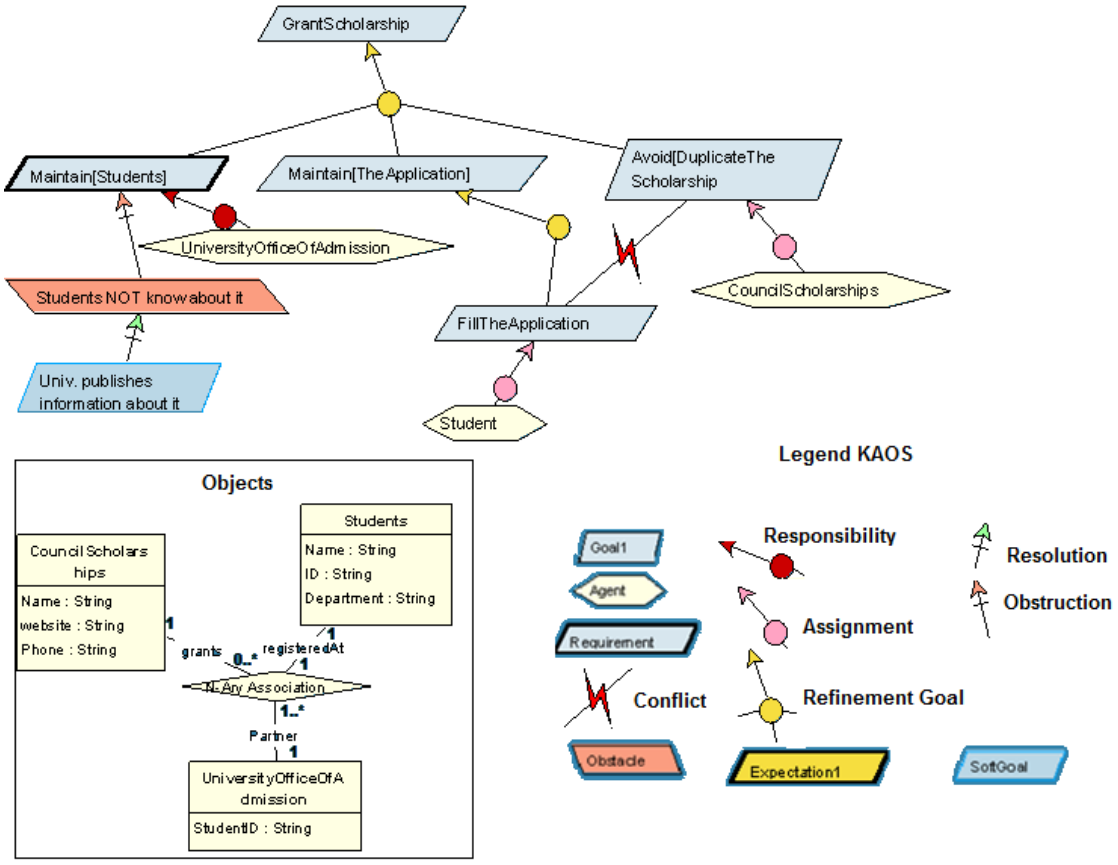

Figure 4. 1 KAOS Goal model.

Figure 4.1 shows an example of a simplified KAOS model. Goals are related to sub goals through goal refinement links. As an example, the figure shows that the goal *Grant Scholarship is* refined into the two conjoined sub goals *Maintain [TheApplication]*, *Avoid [DuplicateTheScholarship]* and one Requirement *Maintain[Students]*, *Maintain [TheApplication]* refined into the sub goal *MeetTheCriterion*. Agents are active objects, that is, they are capable of performing operations. Such agents may be software agents, systems, humans. Yellow circles represent refinements of a parent goal, Pink circles are used for expectation assignment to some agent and for Red circles are used for expectation responsibility to some agent. A potential conflict among goals is represented by red flash icon on a link connecting them. The *DuplicateTheScholarship* and *FillTheApplication* are potentially conflicting there; the applications submitted by students who did not receive scholarships are looked into, while those submitted by students who are benefiting from scholarships are dismissed. An obstacle occurs here in the fact that there would be students who haven't heard of said scholarship because there are no information available on it on the university's website or even advertisements stating that applications are being accepted for a scholarship raising *Obstacle[StudentsNOTknowAbout]*. To avoid this obstacle, a method by which the probability of the obstacle occurring is minimised has to be founded through *goal[Univ. publishes information about it]*, which will restrain students from receiving the scholarship. The university will be publishing information on the scholarship as well as sending this information via email to current students. Object model it provides the concept definitions used by the goals *(GrantScholarship and FillTheApplication)*, Agents (*Student*, *CouncilScholarships, UniversityOfficeOfAdmission*).

KAOS supports using semi-formal and linear formal specification language Linear Temporal Logic (LTL) to describe Goals, Obstacles and to perform logical proofs, which gives accuracy and reveals ambiguities. This is what sensitive and critical systems are in need for, which integrates between safety and security after identifying the requirements specifications of both and later reduced to formal languages that reveals complications resulted from achieving the goals of safety and security. Formal specifications can aid in correct design of system requirements specifications and improve the quality of system-to-be [Nakagawa et al., 2007].
The semantic language of KAOS is necessary to ensure the correctness of the safety-critical requirements specifications described for developing the systems.
KAOS semi-formal languages by using restricted natural language and formal language by using temporal logic language.

4.1.1 Semi-Formal Specification Language

In order to get the highest quality of requirements, a set of tools are used during the development life cycle of the system-to-be; semi-formal is one of these tools. Semi-formal is the use of the natural language in the description but in a narrow context relying on terminologies that suit the domain of the system to be developed using "if conditional" and "Boolean logic". (Table 4.1) summarises semi-formal.

We will be using semi-formal on *DuplicateTheScholarship* goal in example (Figure 4.1).

Avoid[*DuplicateTheScholarship*]: [***If*** submit one application ***then***] ***always not*** Rejected

Semi-formal appears to be giving a description of how the "*Avoid*" goal will be achieved.

Table 4. 1 Semi-Formal Language, Adapted from [Lamsweerde, 2009; Traichaiyaporn, 2013].

Achieve goals
Semi-Formal
Achieve[TargetCondition]: [**If** CurrentCondition **then**] **eventually** *TargetCondition*
Obstacle by negating Achieve goal [CurrentCondition **and**] **always not** *TargetCondition*
Another form from Achieve goals by **Cease a target condition**
Semi-Formal
Cease[TargetCondition]: [**If** CurrentCondition **then**] **eventually not** *TargetCondition*
Maintain goals
Semi-Formal
Maintain[GoodCondition]: [**If** CurrentCondition **then**] **always** *GoodCondition*
Avoid goals
Semi-Formal
Avoid[BadCondition]: [**If** CurrentCondition **then**] **always not** *BadCondition*

4.1.2 Formal Specification Language

KAOS enables us from using formal languages to convert requirements specifications into linear temporal logic formulas (also called *truth function)* that are implemented on the goal. (Table 4.2) summarises formal. The following notations for TLT referencing are used (Figure 4.2).

Table 4. 2 Semi-Formal Language, Adapted from [Lamsweerde, 2009; Traichaiyaporn, 2013].

Achieve goals
Formal
CurrentCondition $\Rightarrow \Diamond$ TargetCondition CurrentCondition $\Rightarrow$ º TargetCondition
Obstacle
Formal
CurrentCondition $\Rightarrow \neg \Box$ TargetCondition
Another form from Achieve goals by **Cease a target condition**
Formal
CurrentCondition $\neg \Diamond$ TargetCondition CurrentCondition $\neg$ º TargetCondition
Maintain goals
Formal
CurrentCondition $\Rightarrow$ GoodCondition
Avoid goals
Formal
CurrentCondition $\Rightarrow \neg$ BadCondition

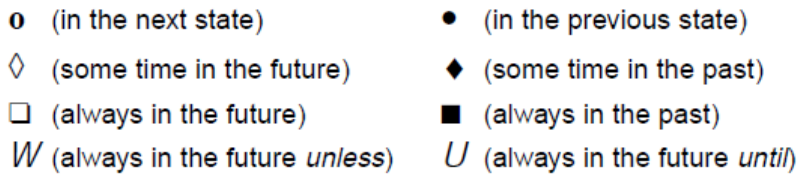

Figure 4. 2 Classical operators for TLT, from [Lamsweerde and Letier, 2000].

We will be using semantics from LTL operators (Figure 4.2) on *GrantScholarship* main goal in example (Figure 4.1).

Goal *Achieve* [*GrantScholarship*]
Concerns St_{SPEC}, App_{SPEC}, Dup_{SPEC}
RefinedTo Students, TheApplication, DuplicateTheScholaship
FormalDef $\forall$ St: Submit, app: Application $\Rightarrow \Diamond \exists$ a: student send (a,App) $\wedge \bullet$[Available (a) $\wedge \neg$Duplicate(a)]

Formal languages have precise notations based on mathematical concepts that work on revealing ambiguity around requirements. However, the existing formal languages in KAOS still need to be improved and enhanced.

One of the challenges that face us is the need for a rigorous semantics specifications language when integrating the requirements specifications of safety and security requirements specifications, which will have a positive impact on reducing the complexities of requirements specification, that were derived using anti-goal and obstacle notations. As pointed out by Matulevičius [Matulevičius, 2008], several experiments were conducted on the use of KAOS elements and narrowing them to B specifications [Matoussi et al., 2009], VDM++ [Nakagawa et al., 2007], A-LTL for Adaptive Systems [Brown et al., 2006] , and another work about modeling correct safety requirements using KAOS and Event-B done by [Traichaiyaporn, 2013].

4.2 KAOS for Security

This section addresses KAOS for security as well as artefact security threat (*Threat obstacles* element). For security requirements analysis and elaboration by the use *Goals* KAOS element, the goal notion allows the expression of security requirements patterns in terms of anti-goals notion and vulnerabilities of the system that is being studied. These patterns can also include a definition of the solution, or counter measure, to the attack in terms of goals that avoid a given vulnerability.

4.2.1 Running example - Security Side

The example *@RemoteSurgery* that was mentioned previously in section 1.2 and ISSRM Risk management process introduced in section 3.2 containing six steps and implement them on the example using KAOS legend goal modelling language (Figure 4.1).

(a) **Context and asset identification**

This step is done through the definition of goals and their refinement in the KAOS goal model, as depicted in (Figure 4.3**)** The main goal studied in the example is Achieve[Record Confidentiality], which is refined in the context domain property *DoctorsWorkingForRemoteSurgery* and the sub-goals *AccessMedicalRecord* associated to the agent *RemoteDoctors, SharingMedicalRecord* associated to the agent *LocalDoctors* and *ReadRecordByAuthorisedDoctors*. More details about the IS are given in the operation model *SharingMedicalRecord*.

The goal *SharingMedicalRecord* is associated to the agent *LocalDoctors.* He also performs other operations (*Select Date StartAndEndSharing, Select RemoteDoctors* and *Select MedicalRecord).* The objects are used to support goals*,* here object is *DatabaseOfDate, NameDoctors and MedicalRecord.*

(b) **Determination of security objective**

Figure 4.3, the determination of security objectives is done in the same model and generally in the same time as the elicitation of other goals. *ReadRecordByAuthorisedDoctors* is an example of security objective; Security need, meaning that we need the CIA of MedicalRecord; HealthcareRecord; PatientData.

(c) **Risk analysis and assessment**

We elaborate security threat by negating the goal *ReadRecordByAuthorised* to obtain the main Obstacles *UserNameVeryWeak* and *PasswordVeryWeak* (Figure 4.4). We elaborate by Obstacle Security Threat analysis to refined the main Obstacles to one sub-obstacle *UseSocialEngeeringToLearnPassword* (Figure 4.4). To operationalisation the obstacle *UseSocialEngeeringToLearnPassword*, we convert it to *Anti-requirement* and assigned to anti-agent *Malevolent* (Figure 4.4).

(d) **Risk treatment**

Risk treatment is defined through the countermeasure chosen for handling the *security obstacle*, and its associated vulnerabilities, obstacle and anti-goals (Figure 4.4; Figure 4.5). In our example, the countermeasure chosen is *Avoidance risk*.

(e) **Security requirements definition**

To avoid the main obstacle *UseSocialEngeeringToLearnPassword*, new anti-obstacle goals are emerging from this countermeasure. A new goal model is thus built, with additional security goal(s), requirement(s). Resolution obstacles by Introduce a new goal Avoid *[AccessToSystemByUnauthorised*] as a countermeasure, this goal refined to into one requirement *PerformAwarenessTraining* (Figure 4.5). This requirement is assigned to the *Security officer* agent.

(f) **Control selection and implementation**

The update of the goal model, which might include the refinement and the operationalisation of the new added avoid goals, constitutes the new system-to-be, as in (Figure 4.5).

This subsection outlines the elaboration of security requirements for the *@RemoteSurgery* system with KAOS.

Elaborating Security Requirements with KAOS, We will elaborate security requirements that are typical in the security domain; CIA. All security goals are expressed in terms of the stakeholder's language**.** This reflects the fact that these are high level goals and are applicable to any alternative design chosen for the system.

Goal Maintain[Security]
InformalDef The system is secure
Category SecurityGoal

Goal Maintain[Integrity]
InformalDef information is guarded against unauthorised update
Category SecurityGoal
Refines Security

Goal Maintain[Confidentiality]
InformalDef information is guarded against unauthorised disclosure
Category SecurityGoal
Refines Security

Goal Maintain[Availability]
InformalDef information is guarded against disruption of service
Category SecurityGoal
Refines Security

Name PerformAwarenessTraining
InformalDef Start awareness training for doctors
Pattern Achieve
Category Security
FormalDef /

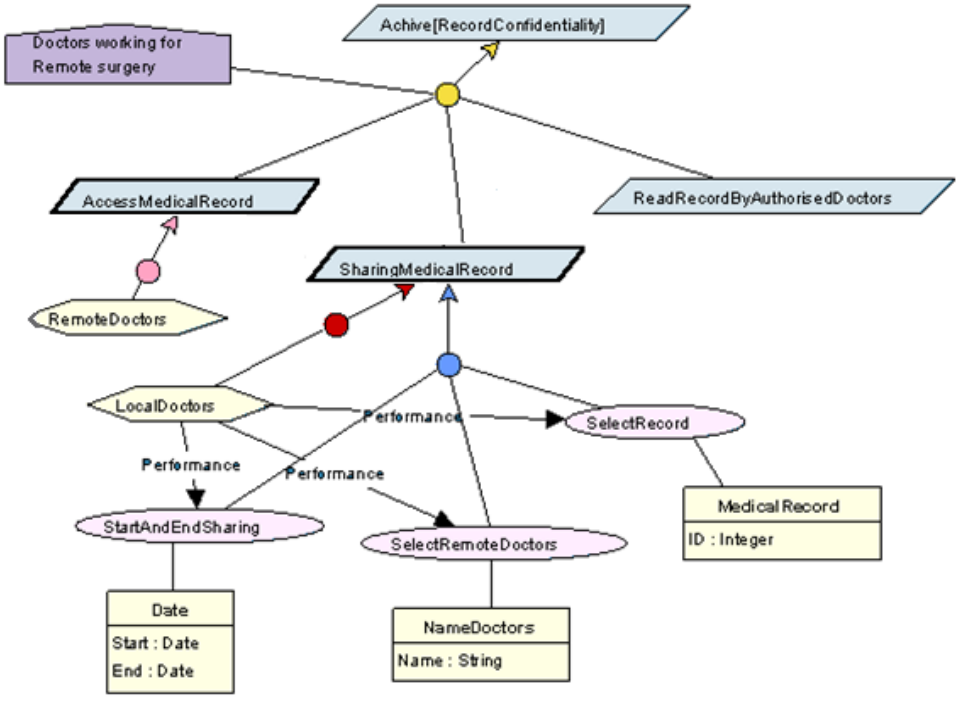

Figure 4. 3 Asset and security objective modelling in KAOS.

Figure 4. 4 Security Obstacle Threat Risk analysis.

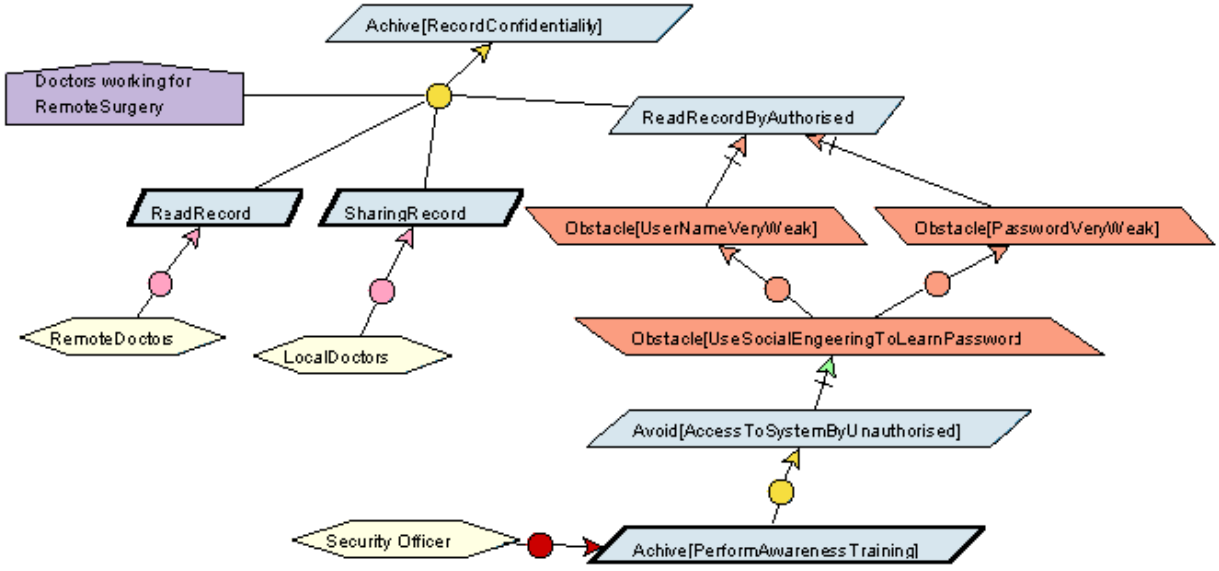

Figure 4. 5 Security requirements and control modelling in KAOS.

4.2.2 Alignment between KAOS and ISSRM domain model.

It was not flexible to build a model for anti-requirements or anti-goal using *obstacles*. We should have converted these *obstacles* into operationalize model by using the *requirements* element and pairing them with *agents* and finally adding *operation* elements.

Concepts related to ISSRM asset are represented by KAOS goal, requirement and expectations. Operation and object are used to present asset while security criteria are represented by goal and object attributes. Threat agent is presented by anti-agent while action method is represented by operationalization, domain and required conditions and operation. Vulnerability is defined by the domain property. At a higher abstraction level, anti-goal represents event while it represents threat in lower levels (in combination with anti-requirements and anti-expectation). Security requirement is represented by security goal. This goal can be refined further by security requirement and expectation [Mayer, 2009].

Discussion about the name of the concepts included in the ISSRM model

Details on harmonisation between the concepts of KAOS and the domain model ISSRM (see Table 4.3) can be found in [Mayer, 2009] on three levels: *Asset-related concepts, Risk-related concepts, Risk treatment-related* concepts in details.

Table 4. 3 Concepts alignment between KAOS extended to security and the ISSRM domain model.

ISSRM domain model		**KAOS extended to security**		Elements of the example (Section 4.2.1)
		Synonyms in [Lamsweerde, 2004]	Language concept (modelling construct)	
Asset-related concepts	Asset	Asset	Goal, Requirement, Expectation, Operation, Object	Goal[*AccessMedicalRecord*] Goal*[SharingMedicalRecord]* Goal*[ReadRecordByAuthorisedDoctors]*
	Business asset			
	IS asset			
	Security criteria	Security Goal	Goal	Avoid[*ReadRecordByAuthorised]*
Risk-related concepts	Risk	/	/	/
	Impact	/	/	/
	Event	ThreatObstacle; anti-goal	Goal, Requirement, Expectation (in anti-model)	ThreatObstacle[*UserNameVeryWeak*] ThreatObstacle[*PasswordVeryWeak*]
	Threat			
	Vulnerability	Vulnerability, domain property	Domain property	*DoctorsAreNotSecurityAware*
	Threat agent	Attackers, malicious agent, anti-agent	Agent	*Agent[Malevolent*]
	Attack method	Potential capabilities of the attacker	Operationaisation + Domain and required conditions + Operations	*Operationalisation [UseSocialEngeeringToLearnPassword*]
Risk treatment - related concepts	Risk treatment	Countermeasures	/	Avoidance
	Security requirements	Security goal, security requirement, security expectation	Goal, Requirement, Expectation	Achieve*[PerformAwarenessTraining*]
	Control	/	New model implementing security components.	/

4.3 KAOS for Safety

This section addresses KAOS for safety as well as artefact *Safety Obstacles* (Hazard).
In software engineering, requirements specifications are documents that describe what a system has to perform in order for the stakeholders needs from a new software system to be met.
For safety requirements, it is very important to deal with *Obstacles* (hazard) KAOS element, which capture undesired properties. It allows analysts to identify and address exceptional circumstances during requirements engineering in order to produce robust or new requirements to avoid or reduce the impact of obstacles giving more reliable software [Lamsweerde and Letier, 2000].
The more specific the goal is, the more specific its obstructing obstacles will be. As mentioned earlier, a high-level goal produces high-level obstacles that will be refined into much smaller sub-obstacles. These sub-obstacles are used for precise obstacle identification in order to evaluate their feasibility through agent behaviour negative scenarios. It is much easier and preferable to refine what is wanted than what is not wanted.
The level of how extensive obstacle identification is depends on the type and priority of the obstructed goal. For example, obstacle identification in Safety Goals needs to be adequately extensive. Domain-specific cost-benefit analysis needs to be performed to decide when the obstacle identification process should terminate.
Obstacle OR-refinement yields sufficient sub-obstacles to establish the obstacle; each OR-refinement of an obstacle obstructs the goal obstructed by this obstacle, goals and AND/OR refinement of obstacles proceed exactly the same way except for only a few alternative OR refinements are generally considered, in the case of obstacles, one may identify as many alternative obstacles as possible.

4.3.1 Running example - Safety Side

The example @RemoteSurgery that was mentioned previously in section 1.2 and ISSRM Risk management process introduced in section 3.2 containing six steps, which will be implemented on the example using KAOS legend goal modelling language (Figure 4.1).

(a) Scope and asset identification
This step is done through the definition of goals and their refinement in the KAOS safety goal model, as depicted in (Figure 4.6**)** the main goal studied in the example is *Maintain [AccuracyMovementScale]*, which is refined in the context domain property *SurgeonWellTrained* and the sub-goals *QualityOfImage* associated to the agent *Camara and MinimalLatenc*y. More details about the IS are given in the operation model *QualityOfImage*. The goal *QualityOfImage* is associated to the agent *Camara.* It also performs other operations *(BoundaryDetection and ImageAcuisitions).*

(b) Determination of quality factor objective
Figure 4.6, the determination of safety objectives is done in the same model and generally in the same time as the elicitation of other goals. *MinimalLatenc*y and *QualityOfImage* are an example of safety objective; safety need, meaning that we need the accuracy, robustness and availability of *MovementScale*; *SurgicalManeuvers*; *MinimalLatency*.

(c) Hazard analysis and assessment

We elaborate safety hazards by negating the goal *AccuracyMovementScale* to obtain the root obstacle *WrongMovementScale* (Figure 4.6; Figure 4.7). We elaborate by hazard analysis to refined the main obstacle *WrongMovementScale* to two sub-obstacle *UnwantedMovements* and *NotSmoothAndPreciseSurgicalManeuvers* (Figure 4.7).

(d) Hazard treatment

Hazard treatment is defined through the countermeasure chosen for handling the *safety obstacle*, and its associated vulnerabilities, obstacles (Figure 4.7). In our example, the countermeasure chosen is *prevent hazard*; controlling and interacting with hazards so they do not become accidents.

(e) Quality requirements definition

Obstacles prevention by introduce a new goal Avoid *WrongMovementScale* as a countermeasures, this goal refined to into two requirement *WantedMovements* and *SmoothAndPreciseSurgicalManeuvers.* This requirements are assigned to the *ComputerSoftware* agent. It also performs operation *FilterOutHandTremors*. (Figure 4.8).

(f) Constraint selection and implementation

The update of the safety goal model, which might include the refinement and the operationalisation of the new added *Achieve* goals, constitutes the new system-to-be, as in (Figure 4.8).

This subsection outlines the elaboration of safety requirements for the *@RemoteSurgery* system with KAOS.

Elaborating Safety Requirements with KAOS, we will elaborate safety requirements that are typical in the safety domain. All safety goals are expressed in terms of the stakeholder's language**.** This reflects the fact that these are high level goals and are applicable to any alternative design chosen for the system.

Goal Maintain[Survivability]

InformalDef	@RemoteSurgery to continue to deliver its services to users in the face of deliberate or accidental attack [Romani et al., 2009].
Category	SafetyGoal

Goal Maintain[Accuracy]

InformalDef	@RemoteSurgery Software attributes that demonstrate the generation of results or correct effects or according to what has been agreed upon [Romani et al., 2009].
Category	AccuracyGoal
Refines	Safety

Goal Maintain[Robustness]

InformalDef	@RemoteSurgery can function correctly in the presence of invalid inputs or stressful environmental conditions [Romani et al., 2009].
Category	RobustnessGoal
Refines	Safety

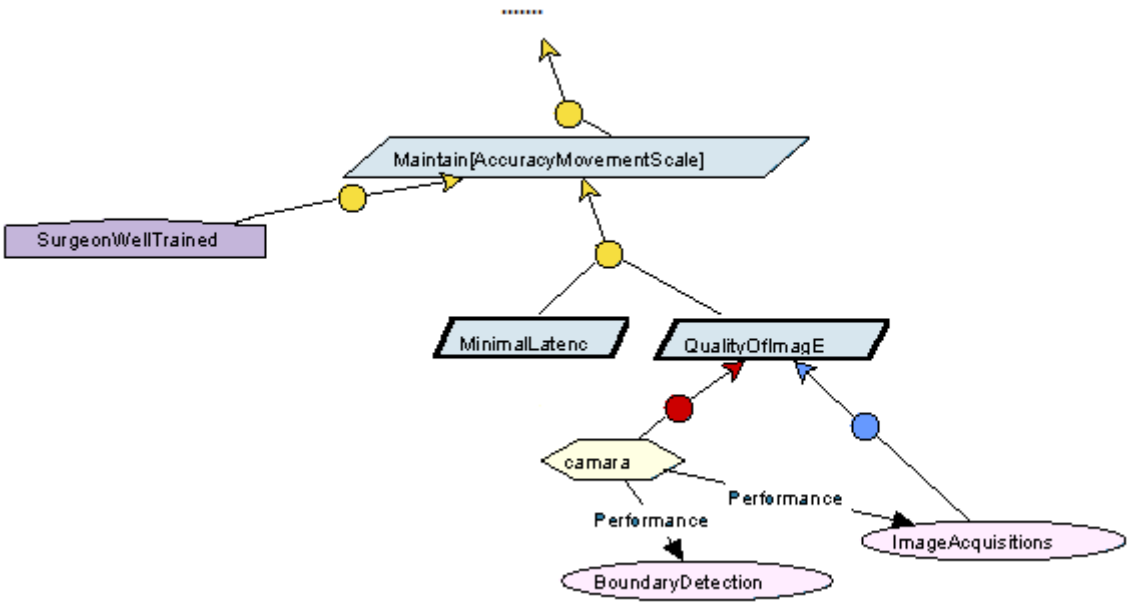

Figure 4. 6 Asset and safety objective modelling in KAOS.

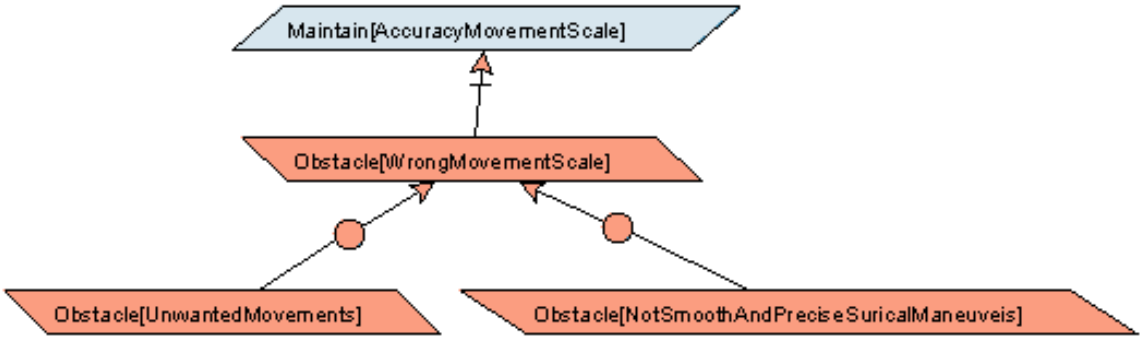

Figure 4. 7 Obstacle Safety Hazard Analysis.

4.3.2 Alignment between KAOS safety and ISRM domain model

In this section we will contribute towards Alignment between KAOS and ISRM and create relationship and mapping between the concepts of both KAOS and ISRM.

Discussion about the name of the concepts included in the ISRM domain model

After identifying the different terms used in each ISRM source, our assumption that the terminology in the ISRM model is not unified has been validated. Many different terms are used to depict the same concept. More than a dozen of different names have been found for some concepts in Table 4.4 (concept (5) and (9)). Sometimes, the same name is used to depict different concepts. For example, Harm is due to an *accident* when dealing with *safety* engineering, is due to an *attack* when dealing with *security* engineering.

The process of extraction and concepts identification (Table 4.4) based on chapter 2 section 2.2 and the definitions used for each concept in the ISRM model.

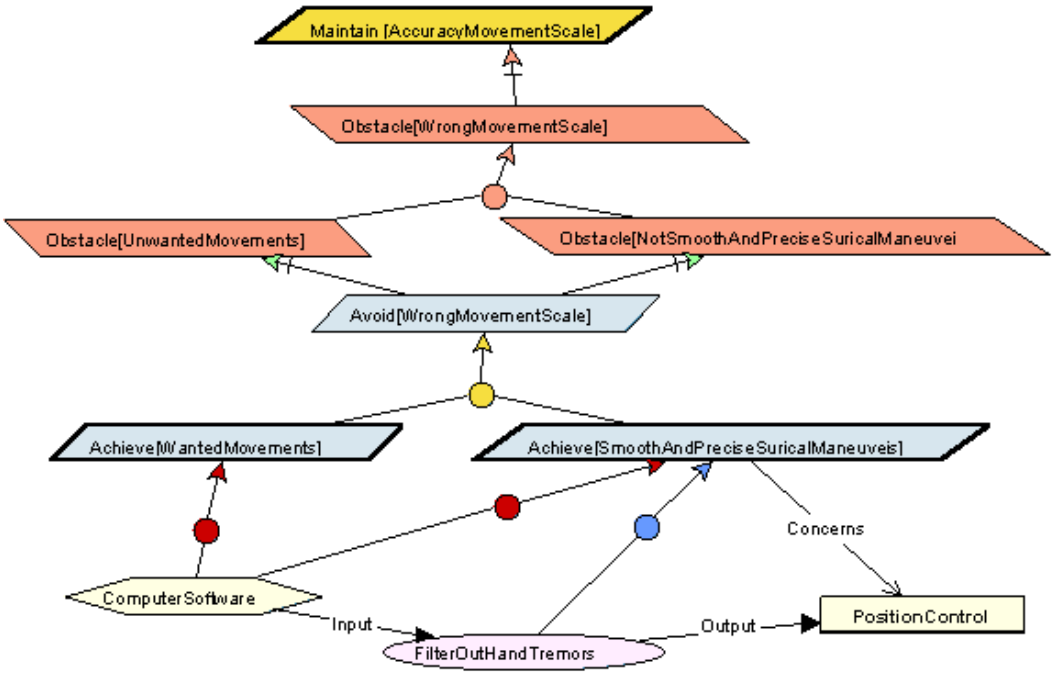

Figure 4. 8 Safety requirements and control modelling in KAOS.

Table 4. 4 Names of the concepts included in the ISRM model.

Type	Concept	Name [Firesmith, 2003; 2009]
Asset-related concepts	1	Asset Systems Environment
	2	Quality subfactor; Safety criteria
Risk-related concepts	3	Risk; Safety risk;
	4	Harm
	5	Danger; Hazard; Accident
	6	Safety vulnerability
Risk treatment - related concepts	7	Safety Goal
	8	Safety requirement
	9	Safety mechanism; Safeguard; Safety tactic

After identifying the concepts comes the aligning process (Table 4.5), and define relationships between concepts of each model.

Table 4. 5 Concept alignment between KAOS extended to safety and the ISRM model.

ISRM model [Firesmith, 2003]		**KAOS extended to safety**		Elements of the example (Section 4.3.1)
		Synonyms in [Lamsweerde, 2004]	Language concept (modelling construct)	
Asset-related concepts	Systems; Environment	Asset	Strategical Goals , Requirement, Expectation, Operation, Object	*Goal [AccuracyMovementScale]* *Goal[MinimalLatency*] *Goal[QualityOfImage]*
	quality subfactor; Safety criteria	Safety Goal	Goal, Object attribute	*Accuracy*
Risk-related concepts	Risk; Safety risk;	/	/	/
	Harm	/	/	/
	Danger; Hazard; Accident	Hazard Obstacle; .	Goal, Requirement, Expectation (in anti-model)	HazardObstacle[*WrongMovementScale*] HazardObstacle[*UnwantedMovements*] HazardObstacle[*SmoothAndPreciseSurgicalManeuvers*]
	Safety vulnerability	Vulnerability, domain property	Domain property	/
Risk treatment -related concepts	Safety Goal	Countermeasures	/	*Prevention*
	Safety requirement	Safety-goal; Safety requirement; Safety expectation	Goal, Requirement, Expectation	Achieve*[WantedMovements];* Achieve*[SmoothAndPreciseSurgicalManeuvers*]
	Safety mechanism; Safeguard; Safety tactic	/	New model implementing security components.	*FilterOutHandTremors*

4.3.3 Discussion about the alignment tables

Asset-related concepts

KAOS is mainly focused on the security of the system-to-be, but it does not make a separation between the IS and business aspects. Thus, we align the Asset ISRM concepts concerning assets with the KAOS *strategical goal*, *requirement and expectation* (Table 4.5). Moreover, their *operationalisation in operation* and *object* are also assets. In KAOS, states of the system-to-be are described using *object attributes*. The purpose of the *Safety goals* is to achieving a target level of safety or one of its *subfactors* (Table 2.2; Section 2.2). In terms of KAOS, this means that the *safety goals* should define *quality subfactor* (Table 2.2; Section 2.2), and *object attributes*, which are concerned by potential risk events

and hazard [Lamsweerde, 2004]. Thus, we align both (safety) goals and object attributes concerned by *goal* with ISRM *quality subfactor*.

Risk-related concepts
In Table 4.5, we align together ISRM *danger, hazard and threat* with KAOS *Negative scenarios*, *anti-goal* (also called malicious *obstacle*). *Anti-goals* can be identified at various abstraction levels, so they might need to be refined until they become *anti-requirements* or *anti-expectations* (assigned to an *anti-agent*). At higher abstraction levels, an *anti-goal* might be considered as the event, which, according to the ISRM model, is a combination of a hazard and one or more vulnerabilities (*safety*). At lower abstraction levels, an *anti-goal* (*anti-requirement or anti-expectation*) is a *hazard*, which is a potential attack or incident to assets. The language concepts for *anti-goal*, *anti-requirement* and *anti-expectation* remains respectively *goal*, *requirement and expectation*.

In Table 4.5, we align ISRM *safety vulnerability* and the KAOS domain property. The KAOS *domain property* is a hypothesis about the domain that holds independently of the system-to-be. In correspondence, ISRM vulnerability (*safety*) is defined as *attributes* of assets. Following the ISRM model, Hazard (*Danger*) cause harm to the assets, due to an *accident* when dealing with *safety* engineering, is due to an *attack* when dealing with *security* engineering.
ISRM domain model does not address hazard agent or hazard method. This explains why in this model, there is no description for agent and operationalization.

Risk treatment-related concepts
ISRM risk treatment corresponds to the *countermeasures* [Lamsweerde, 2004; Lamsweerde and Letier, 2000] that are elaborated after identification of the *anti-goals.* Countermeasures are not KAOS modelling concepts, but rather modelling idioms or patterns adopted by modellers. In KAOS, the countermeasures usually result in new *sefaty goals*, which need to be further refined into realisable *safety requirements* and *expectations*.

In Table 4.5, we align ISRM *safety requirement* and the KAOS *Safety goal (requirements and expectations)*. The refinement and *operationalisation* of the *new safety goals*, their concerned *objects* and *attributes*, and their assignment to *agents (a.k.a software; people; sub-system),* lead to new system-to-be components realising the necessary *safety* means. With respect to the ISRM model, these new system components correspond to *Safety mechanism, Safeguard and Safety tactic* [Firesmith, 2003].

4.4 Summary

We addressed KAOS in the representation of risk management process resulting from the ISSRM domain model in order to represent the security engineering side on the example *@RunningSurgery*. We also worked on the alignments between KAOS and ISSRM domain model and addressing the elements of ISSRM that have been covered in KAOS.

Furthermore, we addressed KAOS in the representation of hazard management process resulting from the ISRM model in order to represent the safety engineering side on the example *@RunningSurgery*. We also worked on the alignments between KAOS and ISRM domain model and addressing the elements of ISRM that have been covered in KAOS.

CHAPTER 5 Common Method to Define Security and Safety (SaS)

"There is no accepted agreement on software safety measures and metric"
[Kornecki, 2003]

We will address the standards and the S*a*S domain produced followed by hazard/risk management process and S*a*S modelling languages. This chapter also includes an introduction on STAMP approach which is expanded upon in the appendices A, B, and C**.**

5.1 Software Safety and Security Engineering Approach and Standards

New standards were created to deal with software-intensive systems [ISO 14971, 2012], cyber-physical systems [IEC 62645, 2011], and shared-control systems [FDA, 2013]. These modern standards define the nature of maintaining (considering its legacy software systems, and connecting these systems to the network is highly risky because they lose the security engineering resistance), or building these systems from scratch to match the requirements of safety and security engineering. Not only that, new laws such as Cybersecurity Act of 2012 [CSA2012] appeared. This bill addresses the threats and weaknesses in critical systems that are connected to the network and trying to take over them.

High-Assurance Cyber Military Systems (HACMS) Clean-Slate Approach was introduced based on the highest quality results for critical systems regarding the safety and security engineering specifications DARPA-BAA-12-21 [DARPA, 2012], through the use of a rigid language for mathematical representation or a semi-automated code synthesis from executable to get formal functions, which are machine-checkable proofed leading to having code that meets with functional specification as well as security and safety specifications. (Figure 5.1) where blue squares represent formal specification, the most important synthesizer component. For a domain-specific, the synthesizer takes the safety and security policies of an element, a functional specification, a description of the target hardware, resource constraints, and the description of the specific environment for the system is to run in.

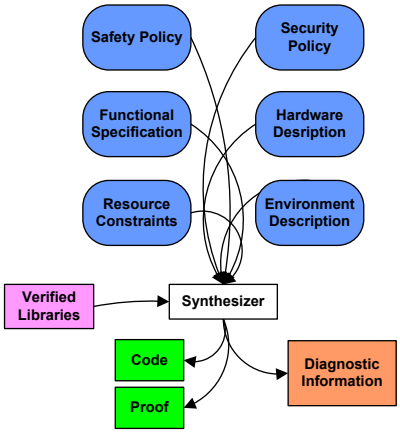

Figure 5. 1 HACMS Clean-Slate Approach, (Adapted from [DARPA, 2012]).

ISO 14971 standard addresses manufacturing medical devices and developing the software for them [ISO 14971, 2012] and aims to integrating the process of risk management and an early stage of design, to produce evidence that their risk assessment process [ISO 14971, 2012] considered and addressed both intentional and unintentional security risks to the medical device with appropriate security controls as part of the device's design. Medical device manufacturers should consider the malicious activity during the early phases of the requirements engineering.

A draft guidance [FDA, 2013] titled "Management of cybersecurity in medical and Hospital Network". The draft discusses the security risks that face medical devices and carry rules to implement safeguards in order to reduce and avoid risk of device failure due to a malicious attack.

5.2 S*a*S Domain Model

S*a*S is a requirement driven software engineering approach is a result of adapting the domain ISRM in chapter 2, which is also a risk analysis approach that inherited the same method from the ISSRM domain model in chapter 3, that deals with security and safety requirements.

S*a*S needs a requirements elaboration method and a design elaboration method in order to cover all the stages of development until implementation is obtained. S*a*S employs the KAOS for eliciting, modeling and analyzing security requirements and safety requirements while it employs the semi-formal specification language and LTL formal specification language for deriving requirements specifications for both safety and security.

The idea behind integrating chapter 3 ISSRM and ISRM domain model in chapter 2 is the main objective that is achieving a certain degree of dependability in the system-to-be. The thrive for achieving dependability in a system-to-be is because the principle of dependability deals with both intentional and unintentional incidents.

This model is the result of merging the ISSRM domain that focuses on security and the ISRM domain that focuses on safety producing a S*a*S domain model (Figure 5.2). Both these domains have been addressed previously.

Discussion of the safety and security model SaS (Figure 5.2):

1. **Assets** *"anything that has value to the organisation and is necessary for achieving its objectives"*. These assets differ from a company to another whether it's (a.k.a. software, IT infrastructure, users, or strategic plan, etc.).
2. **Control** in ISRM [Firesmith, 2003] and ISSRM models [Mayer, 2009], we find that both models agree that there is control that is responsible for meeting the safety and security requirements and minimising the number of vulnerabilities by implementing safeguard and fail-safe methods.
3. **S*a*S Hazard** concept is often used when dealing with systems that if an error occurred, the environment in which the system exists would be affected. The researchers [Chapon et al, 2011] have derived concept for it to describe both safety(ISRM) and security(ISSRM).

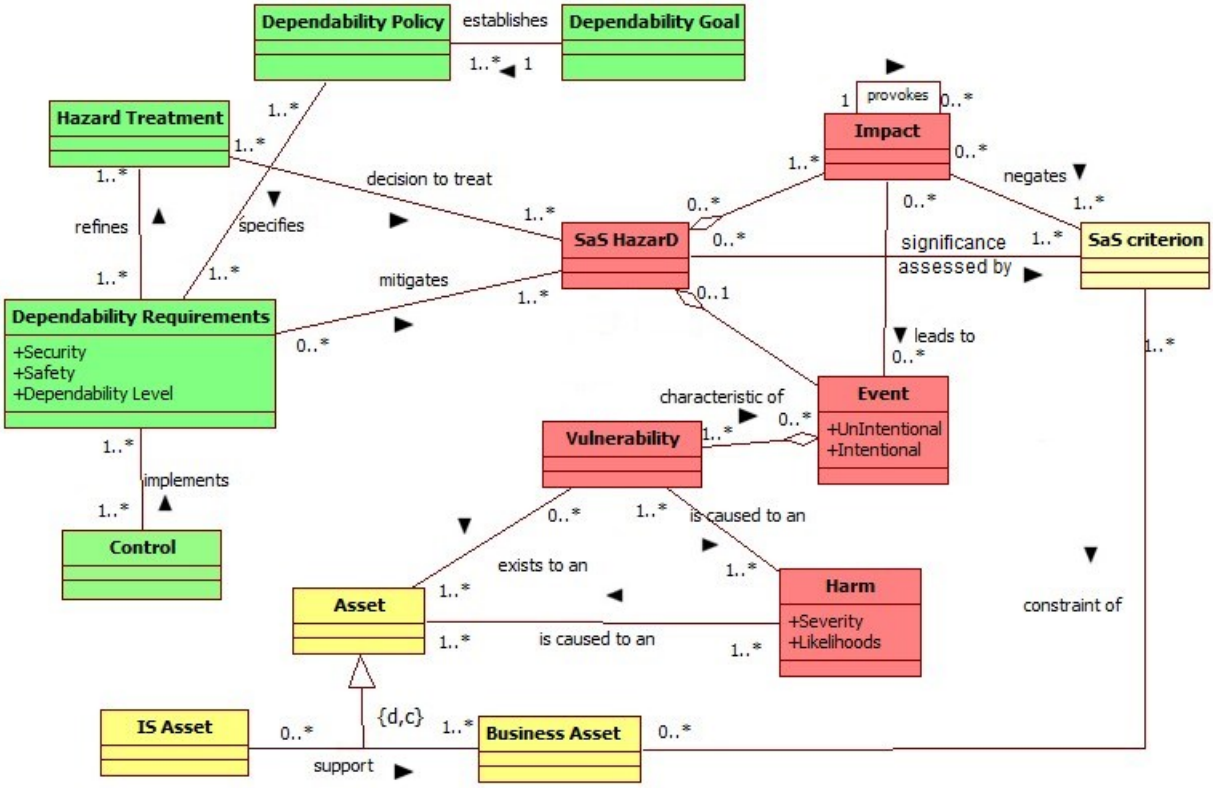

Figure 5. 2 S*a*S domain model.

4. **Event** is the combination of a threat(ISSRM)/hazard(ISRM) and one or more intentional/ unintentional vulnerabilities. *Intentional* is feature of security engineering. There is always a motive and the intention behind planning an attack against the CIA concepts. *Unintentional* is feature of safety engineering. We do not know what accidents we will be facing and so, safety engineering follows very strict mathematical, qualitative, and quantitative methods to accurately analyse risks in order to limit the occurrence of any hazard and the control them.
5. **Harm** is a significant damage, usually associated with an asset that is caused by a hazard, it come from the combination of identified severities and identified likelihoods.
6. **Impact** the latent negative consequence of a hazard, where it negates dependability requirement criteria (Table 5.1).
7. **Dependability requirements** is the umbrella under which come many attributes including those of safety and security. These attributes are chosen based on the nature of the system to be developed and only one either for safety or security might be chosen, or both, according to what the researchers [Avizienis et al., 2004] have addressed in details the concepts and definitions of dependable. Dependability requirement should to be resilience to *Intentional* threats and *Unintentional* hazards.
8. **Hazard Treatment** is the kinds of quality requirements of dependability requirement after choosing with the attributes associated quality characteristics and

quality measures. Standards must be defined according to the system to be developed. From a security perspective, these standards should be CIA standards, or in some cases non-repudiation ones. On the other hand, survivability, quality of service, fault tolerance, correctness, reliability, verification, validation, and maintainability from the safety perspective. It is important to mention that, depending on chosen dependability requirement, the dependability attributes (Table 5.1), and (Table 5.2) show a sub-criterion of performance attribute. The researcher Firesmith has listed the concepts of security, safety, and survivability under a more general concept, defensibility [Firesmith, 2003], which is a special case of dependability.

Table 5. 1 Dependability Attributes of S*a*S, (Adapted from [Firesmith, 2012; Romani et al., 2009]).

Concepts Criterion S*a*S	
Safety	Security
Fail-safe	Confidentiality
Failure tolerance	Integrity
Performance	Availability
Robustness	
Correctness	
Accuracy	
Traceability	
Recoverability	
Human backup	

9. **Dependability policy** are responsible for preventing chosen attributes requirements conflict as it determines the priority for each one.
10. **Dependability goal** the operational level of the system is determined to work in the environment it was built for depending on whether this system will be used by everyone, professionals in a certain field, or a team that was well-trained. This is due to the fact that the product has a very high level of risk in case of human errors. After responsible authorities test the system, in this case it is Civil/Federal Aviation Authority [FAA, 2007; 1998], the final product of the system will be granted Airworthiness Certificate.
    Examples of *Dependability* goals might be, "The *@RemoteSurgery* must be safe" or "*@RemoteSurgery* must be secure".

Table 5. 2 Sub-criterion of performance, (from [Firesmith, 2003b]).

Sub-criterion	Definition
Jitter	*" is the precision (i.e., variability) of the time when one or more events occur."*
Latency	*" is the time it takes to actually provide a requested service or allow access to a resource."*
Response time	*" time is the degree to which the time it takes to initially respond to a request for a service or to access a resource."*
Scheduleability	*" is the degree to which events and behaviors can be scheduled. "*
Throughput	*" is the number of times that a service is provided within a specified unit of time."*

After identifying the different terms used in each S*a*S source, our assumption that the concepts in the S*a*S model is not unified has been validated. Many different terms are used to depict the same concept. More than a dozen of different names have been found for some concepts in Table 5.3.

The process of extraction and concepts alignment (Table 5.3) based on (chapter 2 section 2.2, chapter 3 section 3.1 and chapter 5 section 5.2) and the definitions used for each concept in the S*a*S model.

Table 5. 3 Concepts alignment between ISSRM, ISRM and SaS models.

Type	ISRM model	ISSRM model	S*a*S model
Asset-related concepts	Asset; Systems; Environment;	Asset; Business asset; IS asset;	Asset; Business asset; IS asset;
	Safety criteria; Quality subfactor	Security criteria	S*a*S criteria
Risk-related concepts	Risk; Safety risk; Safety vulnerability; Danger; Hazard; Accident;	Risk; Impact; Event; Threat; Vulnerability; Threat agent; Attack method	Unintentional; Intentional; Hazard; Impact; Event; Vulnerability; Harm
Risk treatment -related concepts	Safety Goal; Safety requirement**;** Safety mechanism; Safeguard; Safety tactic	Risk treatment ; Security requirements; Control	Hazard treatment; Dependability Requirement; Dependability Goal; Dependability Policy; Control

5.3 S*a*S Risk Management Process

A lot of effort was invested in developing a common cross-industry approach to managing risk such as ICH Q9 [ICH, 2005] an recently, ISO 14971 [ISO 14971, 2012], which defines the analysis requirements for medical devices that have the ability to connect to the network from the start of the production process for these devices.
We elicited these six steps [ISO 14971, 2012; ICH, 2005] process (Figure 5.3) for SaS risk management from the safety perspective through adapted ISRM model; chapter 2 [Firesmith, 2003] and security perspective through ISSRM domain model; chapter 3 [Mayer, 2009]. The following Steps are (a to f) Summarised are follows.

(a) **Scope and context asset identification** the process of searching for stakeholders to address the safety and security implications, at the system level and their environments (a.k.a. assets, physical, social) for the purpose of defining the scope. After that, the assets of value for the company as well as the assets related to safety and security engineering need to be identified. The output of this step is the definition of the scope and its relation to the system and the environment and a priority list and rankings of assets to be secured from a safety and security perspective starting with the assets of the highest priority.

(b) **Determination of dependability objectives** in this step, we set a dependability need for every asset identified in the previous step, while each asset has its own characteristics, which requires the identification of Safety/Security goals for each of these assets as summarized in (Table 5.1). The existing attributes in Table 5.1 give a general idea on the attributes of critical systems. However, its not necessary that each system contain each attribute keeping in mind that the more attributes there are in a system, the more it would cost. What is Table 5.1; 5.2 is for illustration purposes only. The full set of attributes are available at [Firesmith, 2004; Avizienis et al., 2004].

(c) **Risk analysis and assessment** the third step consists of the identification of existing and potential hazards that are likely to violate the safety/security goals resulting in accidents. Without doubt, these accidents will cause damage to assets and environment.
Using hazard analysis methodologies such as HAZOP, FTA, AT, FMEA for instance, and using the Scenario, AF, KAOS and misuse-cases from a security perspective. After identification, these hazards are evaluated and the degree of risk is measured using quantitative and qualitative analysis. At this stage, the defining the likelihood of occurrence, defining consequence categories, and risk matrix are produced and the result is full information on these hazards. After that, ALARP principle is implemented to measure the tolerance of each hazard [Redmill, 1999]. If the results are dissatisfying, the entire process has to be performed again starting from step (a), otherwise, the process proceeds to step (d).

(d) **Risk treatment** in this step the decision is made regarding these hazards. These types of risk treatments are divided to three categories: prevention, reducing, or retaining risk.

(e) **Dependability requirements definition** depending on the decision(s) made and choosing the measures in the previous step, we derive the *Control*, and the strategic decision that will satisfy dependability requirement to define SIL target that complies to what has been chosen in order to mitigate and control harms resulting from hazards.

(f) **Constraint selection and implementation** in this step, the decisions made regarding hazards are implemented by setting constraints that comply to SIL target in parallel with implementing safeguards for intentional and unintentional hazards. To ensure the compatibility of the chosen dependability criterion for each asset individually by referring to the dependability policy.

Safety-critical and security-critical software systems are dynamic and interactive resulting in having unintentional hazards. The upgrading process is continuous as the main objective of monitor the residual risk and its compliance to the standards and certificate [Axelrod, 2012; Brazendale, 1995].

5.4 SaS Techniques Modelling

Techniques languages such as Swiss cheese model [Pemeger, 2005] and AcciMap [Salmon et al., 2012] classified as systems-based accident analysis methods. The approaches of these techniques are not domain-specific in accident analysis for a particular industry and what makes these approaches stand out is that the socio-technical aspect is taken into account

during the analysis. Furthermore, these approaches are used in different industries such as aviation, defence, food, public health, oil and gas, and rail transport.

Swiss cheese model, developed by Reason [Reason, 1990], is a well-known even-chain model that uses the swiss cheese metaphor making use of slices in the cheese itself, which represent barriers in a system that intend to prevent errors that could lead to unfavourable events from occurring, and the holes in each slice that suggest multiple contributors. However the swiss cheese model is not without drawbacks and is not accepted uncritically [Preneger, 2005]. The swiss cheese model [Reason, 1990] describes both the interaction between system wide latent conditions and unsafe acts made by human operators and their roles in accidents, and the role of defences and engineering safety features designed to prevent accidents.

AcciMap, an analysis technique based on Ramsussen's risk management framework [Rasmussen, 1997], is a generic approach that does not use taxonomies of failures across the different levels considered and designed specifically for analysing the cause of acidents and incidents that occur in complex socio-technical systems [Rasmussen, 1997; Rasmussen and Svedung, 2000]. AcciMap involves the construction of a multi-layered causal diagram in which the various causes of an accident are arranged according to their causal remoteness from the outcome. The lower levels typically represent the immediate precursors to the event, relating to the activities of workers and to physical events, processes and conditions that contributed to the outcome. The highest levels generally incorporate governmental or societal-level causal factors, which are external to the organisation(s) involved in the event [Salmon et al., 2012]. This way, the full range of factors that contributed to the event is modelled.

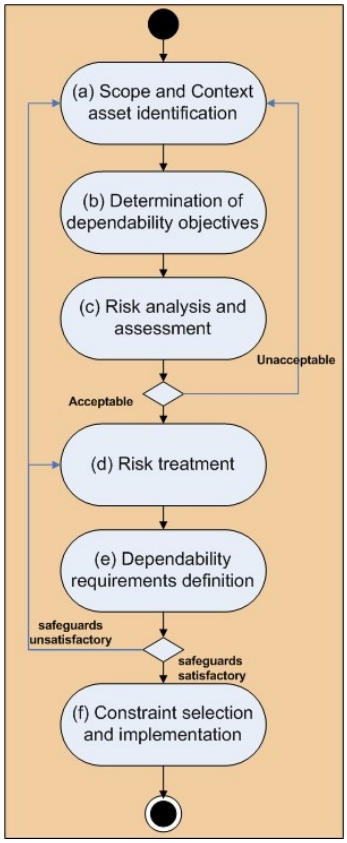

Figure 5. 3 SaS Risk Management Process, (Adapted from [ISO 14971, 2012; ICH, 2005]).

5.5 STAMP Approach

STAMP was developed by Nancy G. Leveson [Leveson, 2004], and is similar in theory with traditional hazard analysis methods (Figure 5.4) (for example, Rasmussen's risk management framework and AcciMap). However, Nancy [Leveson, 2005] considers these traditional hazard analysis methods only to deal with monitoring the accident flow and how it occurred. The traditional methods cannot explain how the accident happened in a component-nested system and taking the socio-technical aspects into consideration, which in turn interacts with this system [Leveson, 2012]. It is considered as a cognitive hazard analysis method because it integrates all aspects of risk, including organizational and social aspects to understand accident causation.

She then published various publications extending the former, focussed on safety [Leveson, 2012; Leveson, 2005]. These publications are finally reinforced by some recent tutorials [Leveson, 2014; 2013; 2012], presenting his work in-depth about safety engineering, which are part of resilience engineering [Hollnagel et al., 2006].

STAMP [Leveson, 2004] is the most recent approach to be developed (Figure 5.4), and considered a new accident causality model based on systems theory [Leveson, 2012]. STAMP approach deals with safety through a technical language called STPA that interacts with identifying hazards and hazard analysis [Pereira, et al., 2006], it can be used early in the system development life cycle to elicitation high level safety requirements and constraints in terms of identifying more causal factors and hazardous scenarios [Leveson, 2012].

Young [Young and Leveson, 2014; 2013] has developed an extension of STPA to serve the hazard analysis for security engineering (Young used term *cyber-security*) and known as STPA-*Sec*. Young published some recent tutorials [Young, 2014] presenting his work in-depth about security (cyber-security).

Some tools that automate the activities of STPA were developed that support the hazard and accident analysis processes including A-STPA[1] and SpecTRM[2].

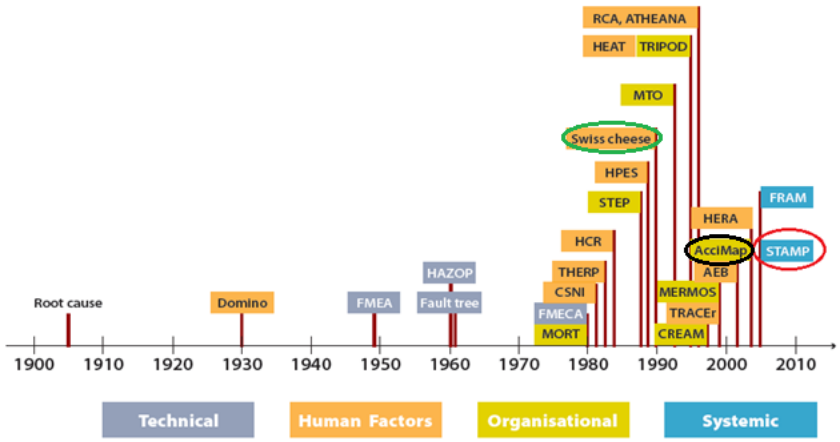

Figure 5. 4 Accident Analysis and Risk Assessment Methods, (From [Eurocontrol, 2009]).

[1] http://www.iste.uni-stuttgart.de/en/se/werkzeuge/a-stpa.html, accessed April 27, 2014.
[2] http://www.safeware-eng.com/software%20safety%20products/features.htm, accessed April 27, 2014.

STAMP approach is explained in details (Appendix A) using SaS information domain model elements as well as the details of running the example "*@RemoteSurgery*" using STPA; *for safety side*, and STPA-*sec*; *for security side* STAMP languages techniques can be found in (Appendices B, and C). The results of this process are discussed in chapter 7.

We will use KAOS language in chapter 6 to represent and run the example *@RemoteSurgery* in the S*a*S side. The results of this process are discussed in chapter 7.

5.6 Summary

We proposed a conceptual domain model through the creation of a (S*a*S) domain model that integrates safety and security domains giving a better opportunity for interplay and integration to find a middle ground between the two domains as well as unifying definitions through their mappings onto the common concepts. We addressed the standards and the S*a*S domain produced followed by hazard/risk management process and S*a*S modelling languages, also includes an introduction on STAMP approach. And the S*a*S domain and hazard management process will be used in running the example *@RemoteSurgery* using KAOS modelling languages from the safety and security sides.

CHAPTER 6 Knowledge Acquisition in autOmated Specification For S*a*S

6.1 KAOS for S*a*S

This section addresses KAOS for safety and security as well as artefact S*a*S *Obstacles* (Hazard and Threat). Here, we will gather all what has been addressed in sections 4.2 (KAOS for Security) and sections 4.3 (KAOS for Safety).

For safety-critical requirements analysis, it is crucial to deal with obstacles KAOS element. The *Obstacles* element is a common element between safety and security (Table 6.1).

Obstacles KAOS element are not only limited to representing *safety goals* but also depends mainly on the system-to-be, its specifications and specific environment (Table 6.1). It is possible to deal with inaccuracy obstacles or non-satisfaction obstacles [Lamsweerde and Letier, 2000]. Knowing the classification domain; *first step from SaS Risk Management Process - (a) Scope and context asset identification,* of obstacles enables and enhances finding suitable treatments.

Table 6. 1 Obstacle Categories, (Adapted from [Lamsweerde, 2009]).

Types of Obstacle	To Represent
Hazard Obstacle	Goal Safety
Threat Obstacle	Security Goal
Dissatisfaction Obstacle	Satisfaction Goal
Misinformation Obstacle	Information Safety
Inaccuracy Obstacle	Accuracy Goal
Unusability Obstacle	Usability Goal

Dependability in the goal *CorrectnessMovementScale* on safety and security overlaps. From a safety perspective, this is due to the hazards that could threaten (*hazardous*) the communications link between the master console that transmits movements for the operation and the slave console that controls robotic arms that operate on the patient by receiving commands from the performing surgeon as well as the software used to control accuracy and measurement. These threats vary between getting these consoles infected by viruses and malware [Deloitte, 2013], which will drastically affect the performance of these devices especially the connection link because if there was a cyber-attack [FDA, 2013] on the service provider for the hospital to perform the operation remotely is a serious problem. This is because there is zero tolerance when it comes to packets loss or high latency, which could have serious consequences in transmitting images to the operating surgeon and latency in commands receipt on the slave console side for the robotic arms performing the surgery resulting in a possibility of harming the patient. Researchers in this field concluded this as well [Marescaux, et. al., 2002], [Anvari et al., 2005; Anvari, 2007] and [Jumpei et al., 2006]. On the other hand, from a safety perspective, the robotic arms and the camera should not stop working all of a sudden without warning.

6.2 Running example - S*a*S Side

The example *@RemoteSurgery* that was mentioned in Sections 1.4 and 1.5 through S*a*S risk management process introduced in section 5.3 containing six steps and implement them on the example using KAOS legend goal modelling language (Figure 4.1).

(a) Scope and context asset identification

This step is done through the definition of goals and their refinement in the KAOS safety and security goal model, as depicted in (Figure 6.1**)** The main goal studied in the example is *Maintain* [*CorrectnessMovementScale*] for both Safety and Security modelling analysis, which is refined in the context domain property *SurgeonWellTrained* and the sub-goals from safety side *QualityOfImage* associated to the agent *Camara and* sub-goals from security side *MinimalLatenc*y associated to the agent *ServiceProvider*.
More details about the IS are given in the *Safety* operation model *QualityOfImage*. The goal *QualityOfImage* is associated to the agent *Camara.* It also performs other operations *(BoundaryDetection and ImageAcuisitions).*And More details about the IS are given in the *Security* operation model *MinimalLatenc*y. The goal *MinimalLatenc*y is associated to the agent *ServiceProvider.* He also performs other operations (offer *High-Bandwidth* communication) (Figure 6.1).

(b) Determination of dependability objectives

Figure 6.1, the determination of SaS objectives is done in the same model and generally in the same time as the elicitation of other goals. *MinimalLatenc*y and *QualityOfImage* are an example of S*a*S objective; S*a*S need, meaning that we need the accuracy, correctness, robustness, integrity and availability of *MovementScale*; *SurgicalManeuvers*; *MinimalLatency*; *InsertionOfMaliciousSoftware; Redundancy Components*.

(c) Risk analysis and assessment

We elaborate safety and security requirements by negating the S*a*S goal *CorrectnessMovementScale* (for Security and Safety goals) to obtain the root Obstacles *NoAvailability* (Figure 6.1; Figure 2.6). We elaborate by hazard analysis to refined the main Obstacle *NoAvailability* to three sub-obstacle; *DriverUnresponsive; CommuicationUnderDDOSAttack; HardwareUnresponsive* (Figure 6.2).

(d) Risk treatment

Hazard treatment is defined through the countermeasure chosen for handling the *Safety and Security Obstacle*, and its associated vulnerabilities, obstacles (Figure 6.2). In our example, the countermeasure chosen is *prevent hazards*; controlling and interacting with hazards so they do not become accidents.

(e) Dependability requirements definition

Obstacles prevention by introduce a new goals Avoid **[***HighLatency*]; *Achieve*[*RedundancyComponents*]*;*Avoid[*FailedSoftware*]*;*Avoid[*InsertionOfMaliciousSoftware*] as a countermeasures. *HighLatency* and *RedundancyComponents* goals refined to into two requirements *RedundancyCommuicationLine* and *Minimal_Latency*, both requirements are assigned to the *ServiceProvider* agent. *FailedSoftware* and *InsertionOfMaliciousSoftware* goals refined to into two requirements *Driverresponsive* and *WatchdogCheck*, both requirements are assigned to the *ComputerSoftware* agent (Figure 6.3).

(f) **Constraint selection and implementation**

The update of the safety goal model, which might include the refinement and the operationalisation of the new added *Avoid* and *achieve* goals to meet our expectations, constitutes the new system-to-be, as in (Figure 6.3). We outlines the elaboration of Safety and Security requirements for the *@RemoteSurgery* system with KAOS.

Elaborating SaS Requirements with KAOS we will elaborate safety requirements that are typical in the safety domain. Security and safety goals are expressed in terms of the stakeholder's language**.** This reflects the fact that these are high level goals and are applicable to any alternative design chosen for the system.

Goal Maintain[Survivability]

InformalDef	The ability of @RemoteSurgery *"computer-communication, system-base application to continue satisfying certain critical requirements(e.g., requirement for security, safety, accuracy and correctness, ...)in the face of adverse conditions*" [Rus et al., 2003; Romani et al., 2009]
Category	Safety&SecurityGoal

And to achieve the main goal, we refined it to three sub-goals: [*Availability; Accuracy; Correctness]*.

Goal Maintain[Availability]

InformalDef	The ability of @RemoteSurgery *"to be in a state to perform a required function under given conditions at given instant of time or over a given time interval , assuming that the required external resource are provided* "[ESA, 2004 ; Romani et al., 2009]
Category	AvailabilityGoal
Refines	Safety&SecurityGoal

Goal Maintain[Accuracy]

InformalDef	@RemoteSurgery *"Software attributes that demonstrate the generation of results or correct effects or according to what has been agreed upon* "[Romani et al., 2009].
Category	AccuracyGoal
Refines	Safety&SecurityGoal

Goal Maintain[Correctness]

InformalDef	@RemoteSurgery "*the degree to which a work product and its output are free from defects once the work product is delivered*" [Firesmith, 2009].
Category	CorrectnessGoal
Refines	Safety&SecurityGoal

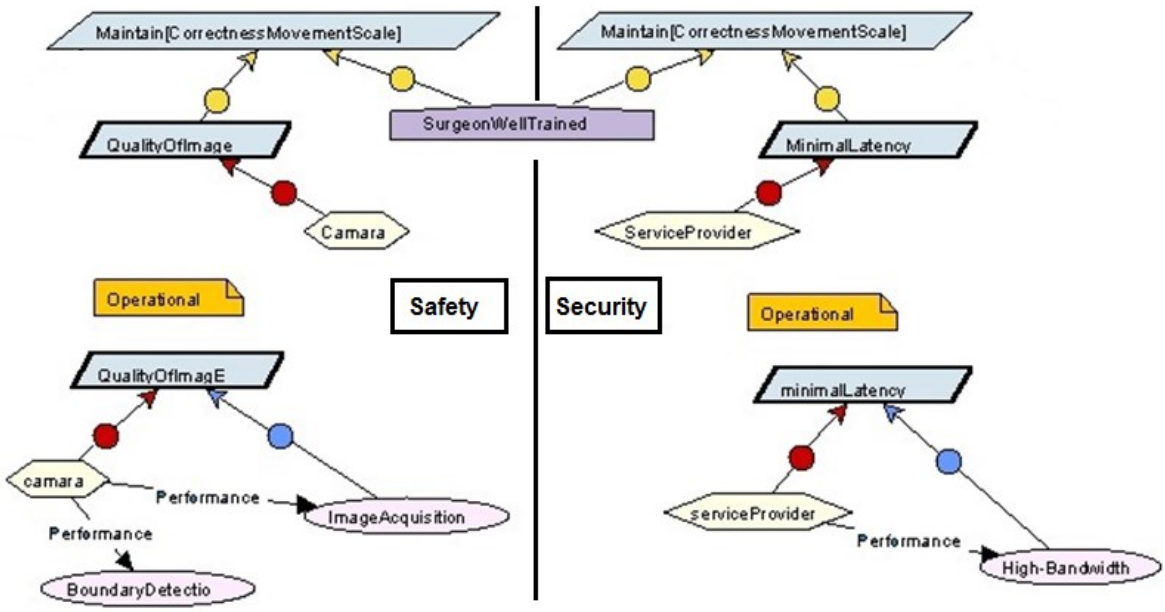

Figure 6. 1 The same asset for safety and security, objective modelling in KAOS.

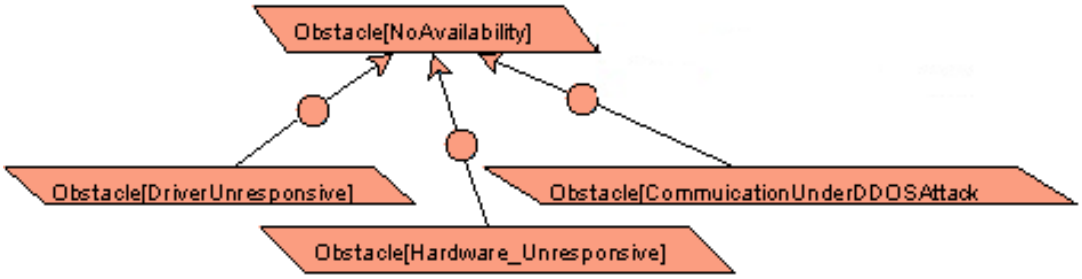

Figure 6. 2 Safety and Security Obstacle Hazard analysis.

6.3 Alignment between KAOS and S*a*S domain model.

In this section we will contribute towards Alignment between KAOS and S*a*S and create relationship and mapping between the concepts of both KAOS and S*a*S.

Discussion about the name of the concepts included in the S*a*S model

After identifying the different terms used in each S*a*S source, our assumption that the terminology in the S*a*S model is not unified has been validated. Many different terms are used to depict the same concept. More than a dozen of different names have been found for some concepts in (Table 5.3).

After identifying the concepts comes the aligning process (Table 6.2), and define relationships between KAOS and S*a*S.

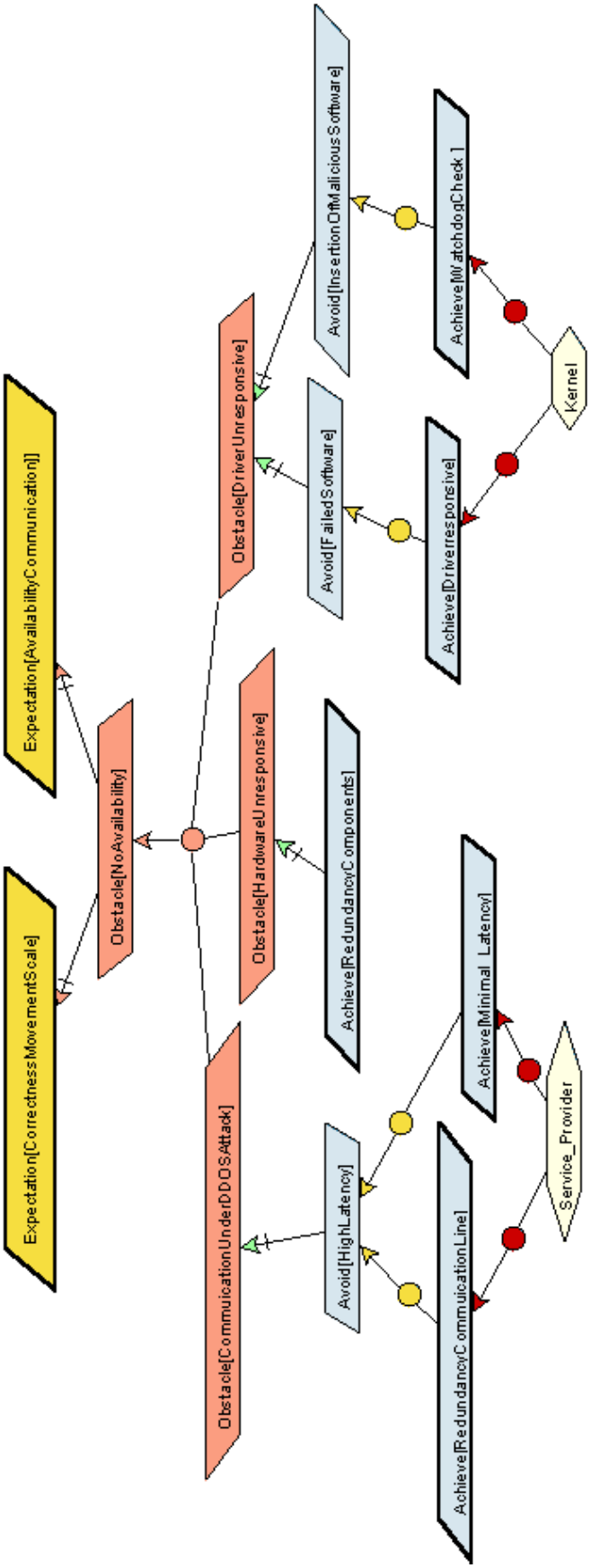

Figure 6. 3 Safety and Security requirements and control modelling in KAOS.

6.3.1 Discussion about the alignment tables

Asset-related concepts
KAOS is mainly focused on the security of the system-to-be, but it does not make a separation between the IS and business aspects. Thus, we align the Asset S*a*S concepts concerning assets with the KAOS *Strategical goal*, *requirement and expectation* (Table 6.2). Moreover, their *operationalisation in operation* and *object* are also assets. In KAOS, states of the system-to-be are described using *object attributes*. The purpose of the *Safety goals* is to achieving a target level of safety or one of its subfactors. In terms of KAOS, this means that the *safety and security goals* should define S*a*S *criterion* (Table 5.1), which are concerned by potential risk events and hazard and/or threat (Table 6.1) [Lamsweerde, 2004]. Thus, we align both (safety) goals concerned by *Expectation; anti-requirements; anti-goals* with S*a*S *requirement criteria*.

Risk-related concepts
In Table 6.2, we align together S*a*S (*unintentional and intentional*) *hazard* (Table 6.1) with KAOS *Obstacle*, *Negative scenarios* (also called *hazard obstacle; threat obstacle; dissatisfaction obstacle; misinformation obstacle; inaccuracy obstacle; unusability obstacle*). *Obstacle* can be identified at various abstraction levels, so they might need to be refined until they become *anti-requirements* or *anti-expectations* (assigned to an *anti-agent*). At higher abstraction levels, an *anti-model* might be considered as the event, which, according to the S*a*S model, is a combination of a hazard and one or more vulnerabilities (safety or security or both or see Table 6.1). At lower abstraction levels, an *anti-model* (*anti-requirement or anti-expectation*) is a *hazard*, which is an *unintentional* attack or *intentional* to assets. The language concepts for *anti-model*, *anti-requirement* and *anti-expectation* remains respectively *goal*, *requirement and expectation*.

In Table 6.2, we align S*a*S *Vulnerability* and the KAOS domain property. The KAOS *domain property* is a hypothesis about the domain that holds independently of the system-to-be. In correspondence, S*a*S vulnerability is defined as *attributes* of assets. Following the S*a*S model, Hazards (Table 6.1) cause harm to the assets, due to an *unintentional accident* when dealing with *safety* engineering, is due to an *intentional attack* when dealing with *security* engineering.
In KAOS, an anti-agent monitors or controls *objects* and their *attributes*, and is thereby capable to hazardous the system-to-be. In (Table 6.2), we align S*a*S *unintentional* and *intentional* and KAOS *malicious agent; non-malicious; anti-agent.* The S*a*S model is not clear is there *attack method* characterises the means by which *intentional* or/and *unintentional attacker* carries out the attack. In KAOS an *anti-agent* performs *operations* that satisfy an *anti-model*. Operations change the state of the system-to-be using input/output relationships over the *objects* and their *attributes*. This means that by performing *operations*, the *anti-agent* (*malicious agent; non-malicious)* breaks the safety and security criteria (Table 5.1) (related to *object attributes*). (Table 6.2), we align S*a*S *unintentional* and *intentional* with the KAOS constructs used to *operationalise the anti-model*, namely *operationalisation*, *domain* and *required conditions* and *operation*.

Risk treatment-related concepts
S*a*S hazard treatment corresponds to the *countermeasures* [Lamsweerde, 2004; Lamsweerde and Letier, 2000] that are elaborated after identification of the *anti-model.* Countermeasures are not KAOS modelling concepts, but rather modelling idioms or

patterns adopted by modellers. In KAOS, the countermeasures usually result in new *dependability goals*, which need to be further refined into realisable *safety and security requirements* and *expectations*. In (Table 6.2), we align S*a*S *security requirements; safety requirement* and the KAOS *Safety goal; security goals (requirements and expectations)*. The refinement and *operationalisation* of the *new safety and security goals*, their concerned *objects* and *attributes*, and their assignment to *agents,* lead to new system-to-be components realising the necessary *safety* and *security* means. With respect to the ISSRM [Mayer, 2009] and ISRM [Firesmith, 2003] information models, these new system components correspond to *Control.*

Table 6. 2 Concept alignment between KAOS extended to SaS.

S*a*S Model		**KAOS extended to S*a*S**		Elements of the example (Section 6.2)
		Synonyms in [Lamsweerde, 2004]	Language concept (modelling construct)	
Asset-related concepts	IS Asset	Asset	Business Goals Strategical Goals , Requirement, Expectation,	Goal [*CorrectnessMovementScale*] for safety and security.
	Business Asset			
	S*a*S criterion	/	Goal verbs, Object attribute	Avoid**[***HighLatency*]; Avoid[*FailedSoftware*]*;* Avoid[*InsertionOfMaliciousSoftware*]
Risk-related concepts	Impact	/	/	/
	Harm	/	/	/
	S*a*S Hazard	Hazard obstacle; Threat obstacle; Dissatisfaction obstacle; Misinformation obstacle; Inaccuracy obstacle; Unusability obstacle	Goal; Goal Safety; Security Goal; Satisfaction Goal; Accuracy Goal; Usability Goal; Requirement; Expectation (in anti-model).	ThreatObstacle*[DriverUnresponsive];* ThreatObstacle*[CommuicationUnderDDOSAttack];* *HazardObstacle[HardwareUnresponsive]*
	Vulnerability	Vulnerability, domain property	Domain property	/
	Event; Unintentional; Intentional	Attackers; malicious agent; non-malicious agent; anti-agent	Agent; Operationaisation	*ServiceProvider(Intentional); Camara (Unintentional);* Unintentiona operationaisation*[QualityOfImage];* Intentional operationaisation*[MinimalLatency];*
	Hazard treatment; Dependability Goal	Countermeasures	/	*Prevent Hazard*
Risk treatment - related concepts	Dependability Requirement; Dependability Policy	Safety-goal, Security-goal, security requirement, security expectation	Goal, Requirement, Expectation	Achieve[*RedundancyComponents*]*;*
	Control	/	New model implementing S*a*S components.	/

The outcome of this work will be compared to that in Appendices B and C in chapter 7.

6.5 Summary

We worked on integrating safety engineering and security engineering through the use of KAOS techniques language, which was used to represent accident management process resulting from the S*a*S domain model in order to represent both the safety and the security sides in the example *@RunningSurgery*. Furthermore, we worked on the alignment between KAOS and S*a*S domain model and addressing the S*a*S elements that have been covered in KAOS.

CHAPTER 7 Validation

"*All models are wrong, some models are useful*"
-George Box

In this chapter, we address the results achieved from this research. Goal, Question, Metric approach (GQM) [Basili et al., 1994] will be used in questioning the metrics used for validation.

The level of maturity of the S*a*S domain model. In other words, we want to count the concepts this domain model inherited from both the ISSRM [Mayer, 2009] and the ISRM [Firesmith, 2003] domains to serve the application of the dependability concept. The metric here is concept completeness.

The second goal is divided into two parts; since S*a*S domain model has inherited several concepts from the ISSRM and the ISRM domain models that overlap, which affected the result of the alignment process between S*a*S concepts and KAOS patterns elements. This is because KAOS does not support constructing redundant elements [Matulevičius and Heymans, 2007]. The metric here is semantic completeness. The second part of this goal is whether the alignment process between KAOS verbs concepts and S*a*S concepts is semantic correctness.

7.1 Case study

To sum it up, a S*a*S information model has been created (chapter 5) and used one of the Goal modeling language (GML) languages, namely KAOS modelling language (chapter 6) to implement the hazard management process for S*a*S on the example *@RunningSurgery*, and the alignment process between the KAOS and the S*a*S domains (chapter 6).

Furthermore, we used STAMP and its STPA-sec modelling language, which are categorised under scenario-based approach. We first align S*a*S and STAMP using the example *@RunningSugery* using a scenario-based approach, then we applied hazard management process from the safety side using STPA on the same example used in S*a*S. The hazard management process was applied from the security side using STPA-*sec*.

7.2 Discussion

we discuss the aforementioned goals. To avoid repeating tables and figures, we will refer to them when necessary.

In table 5.3, we extracted the concepts of the ISSRM [Mayer, 2009], ISRM [Firesmith, 2003], and S*a*S domains then dividing these concepts into three categories; Asset-related concepts; Risk-related concepts; Risk treatment-related concepts. We find that it is clear that the concepts of S*a*S domain are Risk-related concepts are redundant.

The second goal consists of two parts; in Table 6.1, we see that obstacles are divided into six categories. When represented using KAOS, these six categories are reduced to one type

that is later built and can be customized as to which obstacles category it belongs to by adding annotations (Figure 6.2). secondly, there were clear indications that it affected semantic correctness during the alignment process between KAOS elements and S*a*S concepts (Table 6.3). this is due to the fact that each KAOS *patterns* (a.k.a. Avoid, Maintain, Achieve) is met by more than one concept from the S*a*S domain. This applies to the obstacles element since there are six categories (Table 6.1) [Lamsweerde, 2009]. For that, we intended to leave it unexplained using a single term {*Obstacles*} and used an explicit term *ThreatObstacle* that deals with security instead. On the other hand, *HazardObstacles* deals with the safety perspective to support semantic KAOS-S*a*S in our work.

We recall section 6.2 and the appendices B, and C. The results contain differences. These differences are due to two reasons:
The example @RunningSurgery was run using KAOS, which is considered from the GML category. KAOS *patterns* (a.k.a. Avoid, Maintain, Achieve) were used to create obstacle models, and derive milestone from it.

7.3 Cases

We compare the results we got from applying KAOS-S*a*S with the results we have from applying the example *@RunningSurgery* and measure the degree of likelihood using STAMP (STPA; STPA-*sec*).

The same example "*@RunningSurgery* "was run using STPA and STPA-*sec*, which are considered from the scenario-based category that use textual description of the analysis process. Furthermore, the approaches used in the hazard analysis for security (STPA-*sec*) do not differ much from the approaches used in threat analysis for safety. This is clear in the phase "*Identifying unsafe/unsecure control actions*" (Table 7.1) STPA-*sec* as security and safety are inseparable as mentioned by Young [Young and Leveson, 2013; 2014].
This is due to the fact that STPA-*sec* does not take into account traditional security standards like confidentiality, integrity and availability, which leads to ambiguity around the standards to be used when running the example from the security side using STPA-*sec*. however, it has the advantage of being scenario-based because it was helpful using textual description.

Table 7. 1 Summary Steps Risk/Hazard Management process for S*a*S, STPA and STPA-*sec*.

S*a*S	STPA	STPA-sec
1-Scope and context asset identification	1-Identify accidents and hazard	1-Determining unacceptable losses
2-Determination of dependability objectives	2-Construct functional control structure	2-Creating a model of the high level control structure- HLCS
3-Risk analysis and assessment	3-Identify unsafe control actions	3-Identifying unsafe/unsecure control actions
4-Risk treatment	4-Identify causal factors and control flaws	4-Developing security requirements and constraints
5-Dependability requirements definition		5-Identifying casual scenarios
6-Constraint selection and implementation		

7.4 Threat to Validity

Threats of validation varied as follows:

- The modelling language STPA-*sec* is still under development and immature. Furthermore, references are limited to only one tutorial [Young, 2014] and two researches papers that explain the STPA-*sec* processes [Young and Leveson, 2014; 2013]. This shows slight effect when running *@RemoteSurgery* STPA-*sec* security corner.
- The context of the example *@RemoteSurgery* addressed in the study (extracted and built) from the description of three scientific experiments [Marescaux, et. Al., 2002; Anvari et al., 2005; Anvari, 2007; Jumpei et al., 2006] that are affected to a certain level with subjectivity.
- The S*a*S domain model is immature, which affected alignment between it and KAOS.
- Lack of educational papers on the use of STAMP approach. For that, the example *@RemoteSurgery* was simplified during the alignment process between STAMP approach and S*a*S, which has affected the results.

7.5 Lesson Learn

Hazard/Risk management process was the entry point for the integration process as the interface interplays between safety requirements and security requirements from the aspect of system functionality and what the system should and should not do. S*a*S is the result in integrating ISRM hazard management and ISSRM risk management. Were aligned of between KAOS and S*a*S domain model. It became possible to analysis safety and security in a consistent and using one modelling language tool.

Finally, we would like to conclude that using KAOS in this research was suitable to run the experimental researches, and easy to learn as supported by the study conducted by [Matulevičius and Heymans, 2007]. Goal modeling tool, *Objectiver*, was used in the creation of goal models to give contextualization to these goals, and is rich in element shapes [Matulevičius et al., 2006] that supported the use of KAOS.

Chapter 8 Conclusion and Future work

We investigated the available information domain models for safety engineering and found a domain model ISRM [Firesmith, 2003] that addresses safety engineering, which has enriched the understanding of the concepts used in risk management process from the safety engineering aspect. Furthermore, we found the ISSRM domain model [Mayer, 2009] from the security engineering aspect, which has enriched the understanding of the concepts used in risk management process. Therefore, we performed an alignment between KAOS and ISRM domain model concepts [Firesmith, 2003], we performed an alignment between KAOS and ISSRM [Mayer, 2009] domain concepts. This has resulted in extended coverage for the concepts resulting from the integration between safety engineering and security engineering in the risk management process and enlisting all of them in a table.

We used KAOS [Lamsweerde, 2009] that is classified under goal-oriented languages. We have found that KAOS enables a representation method for security and safety hazard/risk management together by used obstacles method.

We propose a solution through the creation of S*a*S information domain model that integrates safety and security domains giving a better opportunity for interplay and integration to find a middle ground between the ISRM hazard management and ISSRM risk management as well as unifying definitions through their mappings onto the common concepts. we performed an alignment between S*a*S domain model concepts and KAOS concepts elements.

8.1 Limitations

our research thesis was met by limitations. The first limitation was the fact that we were talking from a theoretical point of view during the creation of the S*a*S information model. No new metrics were used to enhance the domain, instead, the S*a*S domain inherited the same metrics used in the ISSRM [Mayer, 2009] and ISRM [Firesmith, 2003] domains.

The hazard management process for the S*a*S domain model is one of our research assumptions, used as an input for our research method and is the result of the integration between ISSRM process and ISRM process, which are included in our research assumptions.

The application of hazard management process for the S*a*S information domain model using KAOS modelling language exclusively. Part of the example *@RemoteSurgery* that overlaps with our research was deducted in order to maintain a certain degree of subjectivity.

The STAMP approach is mainly directed to safety engineering in the first place. The STPA modelling language was used in the hazard analysis process from the safety perspective only. The alignment process between concepts of STAMP approach and S*a*S information domain model mainly relied on a simple and specific scenario that was deducted from the example *@RunningSurgery*.
The STPA-*sec* modelling language was used in the hazard analysis process from the security perspective. STPA-*sec* is still under development and validation by Young. The

references are limited to only one tutorial [Young, 2014] and two researches papers that explain the STPA-*sec* processes [Young and Leveson, 2014; 2013]. This shows slight effect when running example STPA-*sec* security corner.

8.2 Conclusion

This research problem is divided into two research questions for investigation. Now we will discuss our answer to these research questions.

RQ1: *How could we possibly relate safety and security?*

To answer this question we have surveyed and analyzed different hazard/risk management process, because it was the entry point for the integration process as the interface interplays between safety requirements and security requirements from the aspect of system functionality and what the system should and should not do. For that, we proposed the creation of an information domain model that integrates between S*a*S, and the implementation of risk management process leading to dependability requirements (safety and security).

RQ2: *What is meant by extend modeling languages approach for safety and security risk management?*

We have investigated alignment between the S*a*S domain models for hazard/risk management with the modeling language KAOS, which has given the possibility for a better method to derive safety and security requirements in early stages from the beginning of the system development life cycle by used obstacles approach. The alignment between S*a*S domain model and KAOS enhances the cooperation and facilitates communication and interaction between stakeholders.

8.3 Future Work

To enhance the process of deriving the requirements from the user, the researcher [Zapata, 2013] proposed a method for weighted salience allocation to the KAOS goals and requirements specification, based only on the hierarchy level, in the context of the UNC-Method. Zapata [Zapata, 2013] concludes that the use of UNC-Method closer to the stakeholder way of thinking. Finally, semantic language of KAOS is applied on UNC-method.

Bibliography

A. Burns, J. McDermid, and J. Dobson, "On the meaning of safety and security," The Computer Journal, vol. 35, no. 1, pp. 3{15, 1992.

A. Pirisi. Telerobotics brings surgical skills to remote communities. The Lancet, 2003; 361(9371).

Abderrahman Matoussi , Régine Laleau , Dorian Petit. 2009. Bridging the gap between KAOS requirements models and B specifications.

Alexander, Ian, "Misuse cases: use cases with hostile intent," Software, IEEE, vol.20, no.1, pp.58,66, Jan/Feb 2003 doi: 10.1109/MS.2003.1159030

Allenby, K.; Kelly, T., "Deriving safety requirements using scenarios," Requirements Engineering, 2001. Proceedings. Fifth IEEE International Symposium on, vol., no., pp.228,235, 2001 doi: 10.1109/ISRE.2001.948563

Andrew J. Kornecki, 2003, Assessment of Software Safety Via Catastrophic Events Coverage.

Anvari M, Broderick T, Stein H, Chapman T, Ghodoussi M, Birch DW, McKinley C, Trudeau P, Dutta S, Goldsmith CH., "The impact of latency on surgical precision and task completion during robotic-assisted remote telepresence surgery", Comput Aided Surg. 2005 Mar;10(2):93-9.

Arata, J.; Takahashi, H.; Pitakwatchara, P.; Warisawa, S.; Konishi, K.; Tanoue, K.; Ieiri, S.; Shimizu, S.; Nakashima, N.; Okamura, K.; Young Soo Kim; Sung Min Kim; Joon-Soo Hahm; Hashizume, M.; Mitsuishi, M., "A remote surgery experiment between Japan-Korea using the minimally invasive surgical system," Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on, vol., no., pp.257,262, 15-19 May 2006.

Avizienis, A., J.-C. Laprie, B. Randell, and C. Landwehr. 2004. “Basic Concepts and Taxonomy of Dependable and Secure Computing.” IEEE Transactions on Dependable and Secure Computing 1(1):11-33.

Axel van Lamsweerde and Emmanuel Letier. 2000. Handling Obstacles in Goal-Oriented Requirements Engineering. IEEE Trans. Softw. Eng. 26, 10 (October 2000), 978-1005.

Axel van Lamsweerde: Requirements Engineering - From System Goals to UML Models to Software Specifications. Wiley 2009, ISBN 978-0-470-01270-3

Axelrod, C.W., "Managing the risks of cyber-physical systems," Systems, Applications and Technology Conference (LISAT), 2013 IEEE Long Island , vol., no., pp.1,6, 3-3 May 2013

Baowei Fei, Wan Sing Ng, Sunita Chauhan, Chee Keong Kwoh, The safety issues of medical robotics, Reliability Engineering & System Safety Volume 73, Issue 2, August 2001, Pages 183–192.

Basili, Victor; Gianluigi Caldiera; H. Dieter Rombach (1994). "The Goal Question Metric Approach".

Benjamin F., Seda G., Maritta H., Thomas S., Holger S., "A comparison of security requirements engineering methods", Requirements Engineering Vol 15, Issue 1 , pp 7-40, Springer-Verlag 2010.

Bieber, P.; Blanquart, J.P.; Descargues, et al. (2012). Security and Safety Assurance for Aerospace Embedded Systems, Proc. 6th Int. Conference on Embedded Real Time Software and Systems (ERTS² 2012), Toulouse, France.

Brazendale John, "IEC 1508: Functional Safety: Safety-Related Systems," Software Engineering Standards Symposium, 1995. (ISESS'95) 'Experience and Practice', Proceedings, Second IEEE International , vol., no., pp.8,17, 21-25 Aug 1995

Brian Davies, Essay: Medical robotics—a bright future, London SW7 2AZ, UK, 22 December 2006.

British Standard BS: IEC61882:2002 Hazard and operability studies (HAZOP studies)- Application Guide British Standards Institution. "This British Standard reproduces verbatim IEC 61882:2001 and implements it as the UK national standard."

C. Warren Axelrod, "Engineering Safe and Secure Software Systems", Artech House; 1 edition (December 1, 2012).

CARLOS MARIO ZAPATA, JUAN FERNANDO ACEVEDO, DAVID MORENO NIÑO, "WEIGHTED SALIENCE ALLOCATION TO GOALS IN THE UNC-METHOD",Dyna rev.fac.nac.minas vol.80 no.180 Medellín July/Aug. 2013.

Committee on National Security Systems (CNSS). National Information Assurance (IA) Glossary (CNSS Instruction No.4009). Fort Meade, Maryland: Committee on National Security Systems (CNSS), National Security Agency (NSA), May 2003.

Common Criteria for Information Technology Security Evaluation, CCMB-2012-09-001, Version 3.1., Release 4 (2012) [Online]. Available http://www.commoncriteriaportal.org/cc/

Council Directive 93/42/EEC of 14 June 1993 concerning medical devices, Official Journal L 169, 12/07/1993 P. 0001 - 0043.

CSA2012, www.hsgac.senate.gov, last accessed on April 10, 2014.

D. G. Firesmith, "Common concepts underly-ing safety, security, and survivability engineering," Technical Note CMU/SEI-2003-TN-033, Carnegie Mellon University, Software Engineering Institute, Dec. 2003.

DARPA, DARPA-BAA-12-21: High-Assurance Cyber Military Systems (HACMS), 2012.

David Peter Eames and Jonathan D. Moffett. 1999. "The Integration of Safety and Security Requirements". In Proceedings of the 18th International Conference on Computer Computer Safety, Reliability and Security (SAFECOMP '99), Massimo Felici, Karama Kanoun, and Alberto Pasquini (Eds.). Springer-Verlag, London, UK, UK, 468-480.

Deloitte, Networked medical device cybersecurity and patient safety: Perspectives of health care information cybersecurity executives, 2013.

Donald Firesmith, "Using Quality Models to Engineer Quality Requirements", Journal of object technology, Vol. 2, No. 5, September - October 2003b

Donald G. Firesmith, "Engineering Safety - and Security-Related Requirements for Software-Intensive Systems". In Proceedings of the 28th International Conference on Software Engineering, ACM SIGSOFT/IEEE, China, pp. 1047-1048.

Donald G. Firesmith, "Engineering Safety- and Security-Related Requirements for Software-Intensive Systems", at the The 11th IASTED International Conference on Software Engineering (SE 2012) in Crete, Greece on 18 June 2012.

Donald G. Firesmith, "A Taxonomy of Safety-Related Requirements," Position paper at the Requirements for High Assurance Systems (RHAS) Workshop at the 12th IEEE International

Conference on Requirements Engineering (RE'2004) in Kyoto, Japan on 6 September 2004, 11 pages.

Erik Hollnagel, David D. Woods, Nancy Leveson, "Resilience Engineering Concepts and Precepts", Chapter 8: Engineering Resilience into Safety-Critical Systems, 2006, ISBN: 9780754649045

ESA, 2004, "Glossary of Terms", ECSS-P-001-B.

Eurocontrol, "A White Paper on Resilience Engineering for ATM", September 2009.

European Space Agency (ESA). Glossary of Terms, Rev. 1 (ECSSP-001A). The Netherlands: European Space Agency, 1997.

FAA Guide For Obtaining a Supplemental Type Certificate; 2007.

FAA, Order 8040.4, Appendix G of the Federal Aviation Administration's June 1998 Safety Risk Management, 1998.

FDA, FDA Safety Communication: Cybersecurity for Medical Devices and Hospital Networks, 2013. Last accessed on April 10, 2014. Available from: http://www.nelsonmullins.com/DocumentDepot/FDA_Cybersecurity_Docs.pdf

Felix Redmill, "An Introduction to the Safety Standard IEC 61508", Journal of the System Safety Society, Volume 35, No. 1, First Quarter 1999

Firesmith D.G., 2009, "Open Process Framework (OPF) Repository Organization (OPFRO) Website". http://www.opfro.org (accessed March 24, 2014).

Great Britain, Ministry of Defence. Safety Management Requirements for Defence Systems (UK MoD Def Stan 00-56). Glasgow, UK: United Kingdom Ministry of Defence, 1996.

Greg Brown, Betty H. C. Cheng, Heather Goldsby, and Ji Zhang. 2006. Goal-oriented specification of adaptation requirements engineering in adaptive systems. In Proceedings of the 2006 international workshop on Self-adaptation and self-managing systems (SEAMS '06). ACM, New York, NY, USA, 23-29.

Håvard Julsrud Hauge, A Survey of Software Safety, Department of Computer and Information Science Norwegian University of Science and Technology, November 23. 2001.

Hiroyuki Nakagawa, Kenji Taguchi, and Shinichi Honiden. 2007. Formal specification generator for KAOS: model transformation approach to generate formal specifications from KAOS requirements models. In Proceedings of the twenty-second IEEE/ACM international conference on automated software engineering (ASE '07). ACM, New York, NY, USA, 531-532.

ICH, QUALITY RISK MANAGEMENT Q9, 9 November 2005.

IEC 62645 Ed.1: Nuclear power plants - Instrumentation and control systems - Requirements for security programmes for computer-based systems, 2011, http://www.iec.ch/dyn/www/f?p=103:52:0::::FSP_ORG_ID,FSP_DOC_ID,FSP_DOC_PIECE_ID:1358,135934,265699

Information technology—security techniques—code of practice for information security management (ISO/IEC FDIS 17799:2005) (2005) International Organization for Standardization.

Information technology—security techniques—management of information and communications technology security—part 1: Concepts and models for information and communications technology security management (ISO/IEC 13335-1:2004)(2004).

INL/EXT-09-17139, Next Generation Nuclear Plant Defense-in-Depth Approach, 2009.

Institute of Electrical and Electronics Engineers (IEEE). IEEE Standard for Software Safety Plans, IEEE-Std-1228. New York, NY: IEEE, 1994.

ISO 14971, "Medical devices. Application of risk management to medical devices", British Standards Institution (BSI), 2012.

J. Daniel, T. Martyn, and M. Lynette I., Editors, Committee on Certifiably Dependable Software Systems, National Research Council, "Software for Dependable Systems: Sufficient Evidence?", 2007.

James Reason, 1990, "The Contribution of Latent Human Failures to the Breakdown of Complex Systems". Philosophical Transactions of the Royal Society of London. Series B, Biological Sciences 327 (1241): 475–484. doi:10.1098/rstb.1990.0090

Klaus M. Hansen, Lisa M. Wells, Thomas Maier,Kai Koskimies, Ludwik Kuzniarz, Johan Lilius, Ivan Porres, .HAZOP Analysis of UML-Based Software Architecture Descriptions of Safety-Critical Systems In Proceedings of NWUML'2004: 2nd Nordic Workshop on the Unified Modeling Language, No. 35. (August 2004), pp. 59-78

Kriaa, S.; Bouissou, M.; Pièvre-Cambacédès, L., "Modeling the Stuxnet attack with BDMP: Towards more formal risk assessments," Risk and Security of Internet and Systems (CRiSIS), 2012 7th International Conference on, vol., no., pp.1,8, 10-12 Oct. 2012 doi: 10.1109/CRISIS.2012.6378942

Kriangkrai Traichaiyaporn, "Modeling Correct Safety Requirements Using KAOS and Event-B", Master's Thesis, September, 2013.

Ludovic Pièvre-Cambacédès, M.Bouissou, "Cross-fertilization between safety and security engineering," Reliability Engineering & System Safety Vol. 110, February 2013, Pages 110–126

Ludovic Pièvre-Cambacédès, T. Quinn, L. Hardin "Cyber Security of Nuclear Instrumentation and Control Systems - Overview of the IEC Standardization Activities" IFAC conference on Manufacturing, Management and Control (MIM 2013), Invited session on Cybersecurity of Control and Safety systems, St. Petersburg, Russia, june 2013

Ludovic Pièvre-Cambacédès and C. Chaudet, "The SEMA referential framework: Avoiding ambiguities in the terms "security" and "safety"," International Journal of Critical Infrastructure Protection, Vol. 3 Issue 2, pp. 55-66, July 2010.

M. B. Line, O. Nordland, L. R stad, and I. A. T ndel, "Safety vs. security?," in Proceedings of the 8th International Conference on Probabilistic Safety Assessment and Management (PSAM 2006), (New Orleans, Louisiana, USA), May 2006.

M. Y. Jung and P. Kazanzides, "Run-time Safety Framework for Component-based Medical Robots," Medical Cyber Physical Systems Workshop (formerly known as HCMDSS (High Confidence Medical Devices, Software, and Systems)), CPSWeek 2013, Philadelphia, PA, USA, 2013

Marescaux J, Leroy J, Rubino F, Vix M, Simone M, Mutter D. Transcontinental Robot Assisted Remote Telesurgery: Feasibility and Potential Applications. Annals of Surgery 2002; 235:487-92

Matt Bishop, Introduction to Computer Security, Addison-Wesley Professional; 1 edition (November 5, 2004), ISBN-10: 0321247442

Matulevičius, R. (2008). Improving the syntax and semantics of goal modelling languages. In: Proceedings of Third International i* Workshop (iStar'08): Third International i* Workshop (iStar'08). (Toim.) Castro, J.; Franch, X.; Perini, A.; Yu, E. CEUR-WS.org, 75-78.

Mayer, N. "Model-based Management of Information System Security Risk." Ph.D. thesis, University of Namur (2009)

Mayer, N., Heymans, P., Matulevičius, R.: Design of a Modelling Language for Information System Security Risk Management. In: Proceedings of the First International Conference on Research Challenges in Information Science, RCIS 2007. pp. 121–132 (2007)

Mehran Anvari, "Remote telepresence surgery: the Canadian experience", Remote telepresence surgery; Telerobotic surgery; Telesurgical networks, Springer-Verlag April 2007, Volume 21, Issue 4, pp 537-541.

Mehran Anvari, Craig McKinley, Harvey Stein, "Establishment of the World's First Telerobotic Remote Surgical Service: For Provision of Advanced Laparoscopic Surgery in a Rural Community" Ann Surg. 2005 March; 241(3): 460–464.

Mouratidis, H., P., Giorgini, P., Manson, G. (2004) 'Using Security Attack Scenarios to Analyse Security during Information Systems Design', Proceedings International Conference on Enterprise Information Systems, Porto-Portugal, pp. 10-17.

Nancy G. Leveson, 2004, A New Accident Model for Engineering Safer Systems. Safety Science, 42(4): 237- 270.

Nancy G. Leveson, Engineering a Safer World: Systems Thinking Applied to Safety, 2012, ISBN: 9780262016629

Nancy G. Leveson, "An STPA Primer Version 1", August 2013b.

Nancy G. Leveson, "Tutorial: Engineering a Safer World", STAMP Conference, 2012.

Nancy G. Leveson, "Tutorial: Engineering a Safer World", STAMP Conference, 2013.

Nancy G. Leveson, "Tutorial: Engineering a Safer World", STAMP Conference, 25 March, 2014.

Nancy G. Leveson, A Systems-Theoretic Approach to Safety in Software-Intensive Systems. IEEE Trans. on Dependable and Secure Computing, January 2005.

National Aeronautics and Space Administration (NASA). Software Safety NASA Technical Standard (NASA-STD-8719.13A). Washington, D.C.: NASA, 1997.

Naved A., Matulevičius R., 2011. Towards transformation guidelines from secure tropos to misuse cases (position paper). In Proceedings of the 7th International Workshop on Software Engineering for Secure Systems (SESS '11). ACM, New York, NY, USA, 36-42.

Nicolas Chapon, Sara Sadvandi, Ludovic Piètre-Cambacédès, "Towards a System Engineering Approach to Master Safety and Security", 23rd ICSSEA 2011 Conference, Paris.

NIST SP 800-26: Security Self-Assessment Guide for Information Technology Systems (2001) National institute of standards and technology

Novak, T.; Treytl, A.; Palensky, P., "Common approach to functional safety and system security in building automation and control systems," Emerging Technologies and Factory Automation, 2007. ETFA. IEEE Conference on, vol., no., pp.1141, 1148, 25-28 Sept. 2007

P. Verssimo, N. Neves, M. Correia, Architecting Dependable Systems, LNCS 2677, Springer, 2003, Ch. Intrusion-Tolerant Architectures: Concepts and Design, pp. 3{36.

Paul M. Salmon, Miranda Cornelissen, Margaret J. Trotter, Systems-based accident analysis methods: A comparison of Accimap, HFACS, and STAMP, Safety Science, Volume 50, Issue 4, April 2012, Pages 1158-1170, ISSN 0925-7535

Piètre-Cambacédès, L.; Bouissou, M., "Modeling safety and security interdependencies with BDMP (Boolean logic Driven Markov Processes)," Systems Man and Cybernetics (SMC), 2010 IEEE International Conference on, vol., no., pp.2852, 2861, 10-13 Oct. 2010 doi: 10.1109/ICSMC.2010.5641922

R.Matulevičius, P.Heymans, G.Sindre. Comparing Goal-modelling Tools with the RE-Tool Evaluation Approach. Information Technology And Control, Kaunas, Technologija, 2006, Vol. 35A, No. 3, 276 - 284.

Raimundas Matulevičius, Patrick Heymans.: Comparing Goal Modelling Languages: An Experiment, Requirements Engineering: Foundation for Software Quality Lecture Notes in Computer Science Volume 4542, 2007, pp 18-32

Rasmussen, J., Svedung. I., 2000, Proactive Risk Management in a Dynamic Society. Karlstad, Sweden, Swedish Rescue Services Agency.

Rasmussen, J., 1997, Risk Management in a Dynamic Society: A Modelling Problem. Safety Science, 27(2/3):183-213.

Raspotnig C., Opdahl A. L. (2013). Comparing risk identification techniques for safety and security requirements, J Systems and Software.

Raspotnig, C. and Karpati, P. and Katta, V. (2012). A Combined Process for Elicitation and Analysis of Safety and Security Requirements, Enterprise, Business-Process and Information Systems Modeling, Lecture Notes in Business Information Processing, vol.113, pp. 347-361, Springer, Berlin.

Respect-IT, "A KAOS Tutorial", 2007.

Romani, M. A. S., Lahoz, C. H. N., and Yano, E. T., “Dependability Attributes for Space Computer Systems”, Proceedings of. 3rd CTA-DLR 'Brazilian Symposium on Aerospace Engineering and Applications/Workshop on Data Analysis and Flight Control” Sao José dos Campos, BR, 2009.

Rune Winther, Ole-Arnt Johnsen, and Bj?rn Axel Gran. 2001. Security Assessments of Safety Critical Systems Using HAZOPs. In Proceedings of the 20th International Conference on Computer Safety, Reliability and Security (SAFECOMP '01), Udo Voges (Ed.). Springer-Verlag, London, UK, UK, 14-24.

Rus, I., Komi-Sirvio, S. and Costa, P., 2003, “Software Dependability Properties: A Survey of Definitions, Measures and Techniques”, Technical Report 03-110, High Dependability Computing Program (HDCP), Fraunhofer Center for Experimental Software Engineering, Maryland, USA, 45 p.

Rus, I., Komi-Sirvio, S. and Costa, P., 2003, “Software Dependability Properties: A Survey of Definitions, Measures and Techniques”, Technical Report 03-110, High Dependability Computing Program (HDCP), Fraunhofer Center for Experimental Software Engineering, Maryland, USA, 45 p.

S. Kriaa, C. Raspotnig, M. Bouissou, L. Pietre-Cambacedes, P. Karpati, Y. Halgand, V. Katta, "Comparing Two Approaches to Safety and Security Modelling: BDMP Technique and CHASSIS

Method", Proc. of the 37th Enlarged Halden Programme Group (EHPG) meeting, Gol, Norway, march 2013

Sommerville, Ian , "An Integrated Approach to Dependability Requirements Engineering", Proceedings of the Eleventh Safety-critical Systems Symposium, Bristol, UK, 4–6 February 2003, pp 3-15.

Steven J. Pereira, Grady Lee, Jeffrey Howard. A System-Theoretic Hazard Analysis Methodology for a Non-advocate Safety Assessment of the Ballistic Missile Defense System, American Institute of Aeronautics and Astronautics, 28 JUN 2006.

Thomas V Perneger, "The Swiss cheese model of safety incidents: are there holes in the metaphor?", 09 November 2005.

Tor Stålhane, Guttorm Sindre, and Lydie Du Bousquet. 2010. "Comparing safety analysis based on sequence diagrams and textual use cases". In Proceedings of the 22nd international conference on Advanced information systems engineering (CAiSE'10), Barbara Pernici (Ed.). Springer-Verlag, Berlin, Heidelberg, 165-179.

Tor Stålhane; Guttorm Sindre, "Safety Hazard Identification by Misuse Cases: Experimental Comparison of Text and Diagrams", 11th International Conference, MoDELS 2008, Toulouse, France, September 28 - October 3, 2008. Proceedings, pp 721-735.

Tor Stålhane; Guttorm Sindre, "A Comparison of Two Approaches to Safety Analysis Based on Use Cases", 26th International Conference on Conceptual Modeling, Auckland, New Zealand, November 5-9, 2007. Proceedings, pp 423-437, 2007.

Ulrich Laible,Thomas Bürger, Günter Pritschow, "A fail-safe dual channel robot control for surgery applications", Safety, Reliability and Security of Industrial Computer Systems, Volume 42, Issue 5, June 2004, Pages 423–436.

Van der Meulen, Meine. Definitions for Hardware and Software Safety Engineers. London, England: Springer, 2000.

Vesely, W. E., Goldberg F. F., Roberts N. H., Haasl D. F. Fault Tree Handbook (1981). Nuclear Regulatory Commission. NUREG–0492.

William E. Young, "STPA-SEC for Cyber Security- Mission Assurance", STAMP Conference, 25 March, 2014.

William Young and Nancy G. Leveson. Systems Thinking for Safety and Security", ACSAC 2013.

William Young and Nancy G. Leveson. 2014. An integrated approach to safety and security based on systems theory. Commun. ACM 57, 2 (February 2014), 31-35. DOI=10.1145/2556938 http://doi.acm.org/10.1145/2556938

Zave P, Jackson M, 1997, Four dark corners of requirements engineering. ACM Trans Softw Eng Methodol 6(1):1–30

Turvaliste ja ohutute süsteemide arendamine KAOS meetodi kasutamisel

Magistritöö (30 EAP)

Mohammed AbuLamddi

Resümee

On spetsiaalsed tehnikad, mida kasutatakse riskihalduses nii turvalisuse kui ohutuse konstrueerimise domeenides. Nende tehnikate väljundid, mida tuntakse artefaktidena, on üksteisest eraldatud, mis toob kaasa mitmeid probleeme, kuna domeenid on sõltumatud ja ei ole domeeni, mis ühendaks neid mõlemat.
Probleemi keskmes on see, et turvalisus- ja ohutusinsenerid töötavad erinevates meeskondades kogu süsteemiarenduse elutsükli jooksul, mille tulemusena riskid ja ohud on ebapiisavalt kaetud.

Käesolevas magistritöös rakendatakse struktuurset lähenemist, turvalisuse ja ohutuse integreerimiseks läbi SaS (Safety and Security) domeeni mudeli loomise, mis integreerib neid mõlemaid. Lisaks töö käigus näidatakse, et on võimalik kasutada eesmärgipõhist KAOS (Knowledge Acquisition in autOmated Specification) keelt ohtude ja riskide analüüsiks, nii et kaetud saavad nii ohutus- kui ka turvadomeen, muutes nende väljundid e. artefaktid hästi struktureerituks, mille tulemusena toimub põhjalik analüüs ja suureneb usaldatavus.

Me pakume välja lahenduse, mis sisaldab sellise domeeni mudeli loomist, milles on integreeritud ohtutuse ja turvalisuse domeenid. See annab parema võrdlus- ja integreerimisvõimaluse, leidmaks kahe domeeni vahelise kesktee ning ühendavad definitsioonid läbi nende kaardistamise üldises ontoloogias.

Selline lahendus toob kokku turvalisuse ja ohutusedomeenide integratsiooni ühtsesse mudelisse, mille tulemusena tekib ohutus- ja turvalisustehnikate vahel vastastikune mõjustus ning toodab väljundeid, mida peetakse usaldusartefaktideks ning kasutab KAOSt domeeni mudeliga, mis on ehitatud juhtumianalüüsi põhjal. Peale vastloodud mudeli rakendumist viiakse läbi katse, milles analüüsitakse sedasama juhtumit, võrdlemaks selle tulemusi teiste juba olemasolevate mudelite tulemustega, et uurida sellise domeeni mõttekust.

Struktureeritud lähenemine võib seega toimida liidesena, mis lihtsustab aktiivset interaktsiooni riski- ja ohuhalduses, aidates leida lahendusi probleemidele ja vastuoludele, mille lahendamiseks on vaja integreerida ohutuse ja turvalisuse domeenid ja kasutada unifitseeritud süsteemianalüüsi tehnikat, mille tulemusena tekib analüüsi tsentraalsus.

Võtmesõnad
Ohutusmudel, turvalisusmudel, usaldatavuse nõuded, eesmärgimudel, eesmärgipõhine modelleerimine, KAOS, infosüsteemide modelleerimine

Appendices

STAMP approach is explained (Appendix A) in details using S*a*S information domain model elements as well as the details of running the example "*@RemoteSurgery*" using STPA; *for safety side*, and STPA-*sec*; *for security side* STAMP languages techniques can be found in (Appendices B, and C). The results of this process are discussed in chapter 7.

We will address the phases of the STPA *process* from the safety perspective (Appendix B), and STPA-*sec process* from the security perspective (Appendix C).

Appendix A Alignment between the concepts of STAMP Approach and SaS domain model

A.1 Explanation STAMP Approach Structure

Figure A.1 shows STAMP a generic hierarchical safety control structure, we address part of the description of *@RemoteSurgery* (Section 1.2) and reflect it on the STAMP domain model.

Using the concepts in S*a*S information domain model (Section 5.2) here to explain STAMP; Scenario-based [Young and Leveson, 2013; 2014] (Figure A.1) in a narrative way.

@RemoteSurgery, A surgeon that is well trained on using the console is considered one of the important assets. Also, the operation itself as well as the patient who will get the operation done using this console. The operation will be performed in a customized environment that meets the standards for operations, which is the hospital and its assets. Without this environment, no operation can be performed without this environment and especially the operations room, which has the necessary tools and the console that the trained surgeon will use to perform the operation (Table A.1).

Table A. 1 Potential summery about STAMP Asset.

Asset	**Reflection**
Organisational asset	Doctors, Operation, Patient
System asset	Surgery consoles, Network connection
Property	Surgery operating room
Environment	Hospital

From a *safety criterion* perspective, it is more about the tools and the environment that have to meet certain standards that comply with MDD, which are as follows: failure tolerance; correctness; accuracy; availability (Table 5.1), and human backup element that comes from *resilience engineering* [Hollnagel et al., 2006]. It is an important factor in this medical field as it is the surgeon that will make a decision and interact manually in case the system becomes out of control.

From a *security criterion* perspective, we are more concerned about the confidentiality of information since it is medical data and being confidential is the normal status. For that, we require CIA (Table 5.1) since confidentiality requires not revealing medical information and treatment costs.

MDD [DIRECTIVE 93/42/EEC, 1993] has divided the medical instruments into four categories depending on the hazard level. The robotic medical instruments are classified under *Class IIb* as shown in (Table A.2).

Table A. 2 MDD divides devices into four classes qualitative scales.

MDD class	Hazard Level
Class I	Low risk,
Class IIa	Medium risk
Class IIb	Medium risk
Class III	High risk

According to MDD, risk analysis and hazard identification must be performed in the design phase following the Drift Correction principle but since the security side is taken into consideration, hazard identification process must be a comprehensive one. The standard IEC 1508 [Brazendale, 1995] confirms performing that as well as SIL; *Reliability-based* identification.

A new method has been found, through research that is currently being used to deal with *unintentional* accident and *intentional* risk in systems that directly interact with the environment, like the new generation of nuclear power plants [INL/EXT-09-17139, 2009]. This method is called Defense in Depth ([Leveson, 2012] used artifact term *Systems Theory*). The systems that use DiD analysis get the results as a preventive plan based on the application of more than one safety layer to face more than one accident. These safety layers are a result of the nature of the system itself. DiD method can be summarized in four essential phases*: Prevention, Control, Protection,* and *Mitigation* respectively. It is important to mention that this analysis will be performed in compatibility with the comprehensive overview specified in *dependability goal* that in turn affects the *dependability policy* that prioritises the requirements in case of *conflict* in *system requirement* ([Leveson, 2012; 2013b] used term *Constraints;* "mean the describe limitations on how the goals can be achieved. But *requirement* mean the behavior required to satisfy the system's goals").

In the phase of *dependability requirement* specification, the execution of both analytic and holistic process and using a different technique, each of the components that interact with the system to be analysed and the interaction with each of these components relying on other components that already exist in the system without separation as well as independent analysis of each of these components. These components are (a.k.a *hardware, software, humans, environment*), which gives us a better overview in dealing with hazards and treatment plans that work with the *dependability goal*.

The process of risk treatment is related to cost, which is the result of analyzing the *dependability goal* phase. There's an inverse relationship between *cost* and *safeguard* requires execution in the system-to-be. The estimated cost resulted from *quantitative* and *qualitative* analysis for both *safety* and *security* requirement (Leveson used term *Constraints*). Theoretically speaking, it is easy to do, but practically, it is very difficult to define the suitable *safeguards* that will be used with safety. For that, we have to keep into account the *safety policy* (this complies to [Leveson, 2012; 2013b] "*Conflicts between goals and constraints can more easily be identified and resolved if they are distinguished*"), that will be used in the system-to-be. These policies are used to comply with the *dependability goal* requirements.

On the top of the component hierarchy pyramid for the STAMP model (Figure A.1), we find congress and legislature, which controls and organizes government regulatory agencies; industry and user associations; insurance companies; unions; and courts. In fact, there are several cases regarding legally allowing the use of *@RemoteSurgery* in hospitals. Furthermore, insurance companies aren't into insuring patients who want to have their surgery performed using *@RemoteSurgery*. Similarly, industry associations are developing training curriculums to train surgeons on using *@RemoteSurgery*, giving them tests to measure their abilities and certify them. User associations affects the patient acceptance or declining the use of *@RemoteSurgery*. STAMP takes into consideration in the sociotechnical cases.

The outcome of this work will be compared to that chapter 6 in chapter 7.

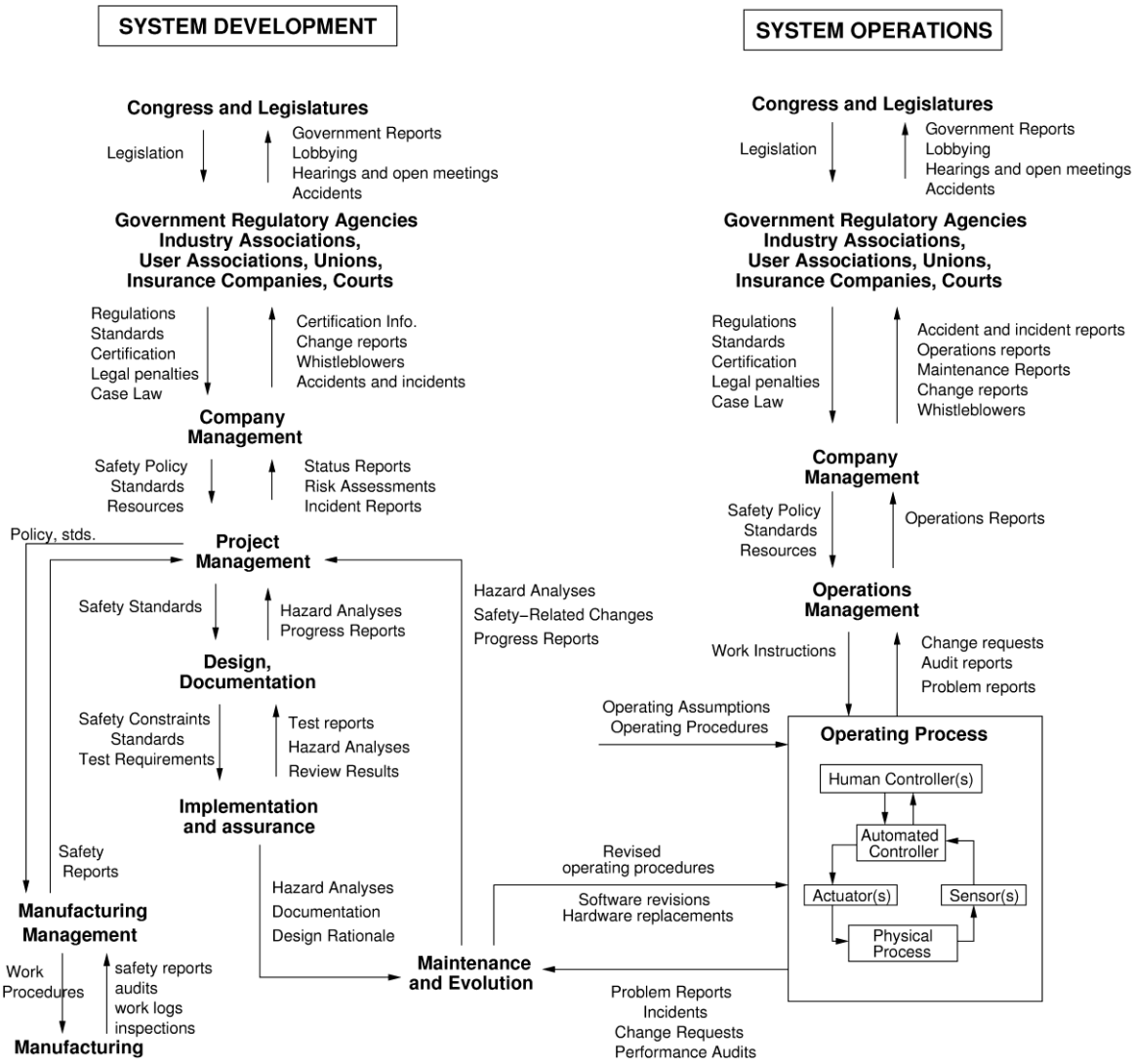

Figure A. 1 STAMP's general generic form of a model of SocioTechnical system control structure. From [Leveson, 2004].

Appendix B STPA Process for Safety

B.1 STPA Process for Safety

The hazard analysis process in STPA [Leveson, 2012] consists of four essential steps (Figure B.1).

a. Identify accidents and hazard

In this step, we work on defining the scope of the system and the environment, defining hazards and potential accidents, and finally, defining the *safety constraints*. This step consists of three sub-steps: *System-level Accidents (Losses); System-level Hazards; System-level Safety Constraints.*

b. Construct functional control structure

In this step, we plan high-level control structure that explains the main components of the system and how it's linked to other components. This would help us understand locations of input and output as well as integration and control in a system.

c. Identify unsafe control actions

In this step, refinement of high-level safety constraints and requirements using scenario-base and the narrative description in defining high-level *unsafe control* actions (Table B.1), which helps us put *safety requirements and constraints* (Table B.2) that are then represented in table form.

Four ways a controller can provide unsafe control (Table B.1): *A control action required for safety is not provided; An unsafe control action is provided; A potentially safe control action is provided too late or too early (at the wrong time) or in the wrong sequence; A control action required for safety is stopped too soon or applied too long*.

d. Identify causal factors and control flaws

We start by identifying causal factors scenarios leading to violation of safety constraints using *control loop model* (Figure B.2) to identify how each potentially hazardous control action identified in Step 3 "*Identify unsafe control actions*" could occur.

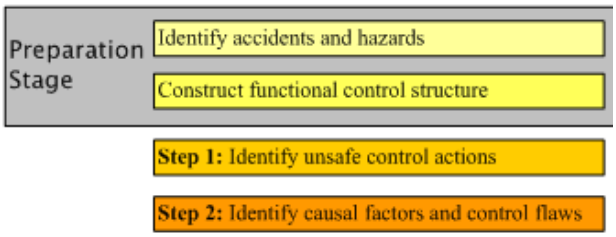

Figure B. 1 Steps System-Theoretic Process Analysis for Safety, adapted from [Leveson, 2012].

Table B. 1 Unsafe Control Actions for example about @RemoteSurgery $_{\text{speed sensor}}$.

Control Action	Unsafe Control Actions $_{\text{speed sensor}}$			
	Not providing causes hazard	Providing causes hazard	Too early/too late, wrong order causes hazard	Stopping too soon/applying too long causes hazard
# **Speed Sensor**	#	#	#	#

Table B. 2 Defining Safety Constraints, for example about @RemoteSurgery $_{\text{speed sensor}}$.

Unsafe Control Actions	Safety Constraint
# **Speed Sensor**	#

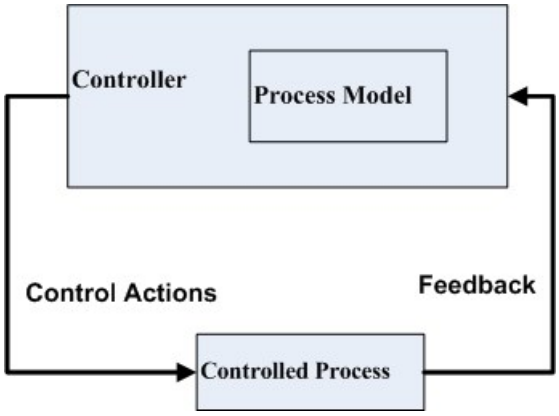

Figure B. 2 High level basic control structure model for Safety, adapted from [Leveson, 2013b].

Therefore, we going to put a simple scenario to explain what we'll be addressing before getting into the hazard analysis process for safety (*STPA*) and security (*STPA-sec*).

@RemoteSurgery, Most of the focus was on the movement of the operating terminals of the main console through which, a surgeon performs the operation. These movements are transmitted from one place to another on the network to reach the receiving console where the patient is, with no latency that might affect the terminals movement. [Anvari et al., 2005] conducted an experiment on remote operations to evaluate the consequences of latency being 500ms [Marescaux et al., 2000] and concluded that the higher the latency value is, the higher the probability of a risk occurring, putting the life of the patient on the line. Furthermore, the transmission of high definition photos allows the operating surgeon to see everything as if they are performing the operation on the patient directly. Avoiding the loss of transmitted packets above the average is also as important as discussed by [Marescaux et al., 2000]. This system, like any other system, is vulnerable simply because it is connected to the network.

The surgeon controls the console; master cart; slave cart, which is a platform that consists of robotic arms for control, cameras, and an operating system that is connected to the host console through a network. This console could be located anywhere in the world and the surgeon sends commands to this console from wherever they are, remotely. The integration between conducting surgeries using robots and platform network connectivity leads us to find a method to identify potential hazard that will stand in the way of this technology that is starting to spread.

In the example, we are dealing with safety/security requirements, where there is no possibility for disconnection; availability between the main console that runs from the surgeon's side and the patient console that receives commands to perform surgery operations and since the system is connected to a network, this puts it under new threats that need to be dealt with.

B.2 Running example- STPA Safety Corner

We apply the four steps of STPA process for safety on the example @RunningSurgery, which was addressed from the safety perspective and KAOS-S*a*S in chapter 6, section 6.2. What is addressed in this section will be addressed using STPA. STPA is considered a base scenario [Leveson, 2004; 2012]. Therefore, we run the example based on the above.

a. Identify accidents and hazard

Here, we have to answer the following questions,

System-level accidents (losses)?
Losing connection with the slave console; Losing control on the console; Losing control on the robot arms
System-level Hazards?
Operating System failure; Software is not responding
System-level safety constraints?
Connection to slave console must not be lost; Control on robot arms must not be lost
Operating software must not fail

b. Construct functional control structure

In this phase, we design the high-level control structure for system-level hazards.
Here, we focus on the analysis in case the robot arms stopped responding to the operating surgeons maneuvers. We have addressed the main components in this analysis according to STPA.

Main components of Figure B.3 are,
*SlaveConsoleCart:*is the platform that received the movements for the robotic SlaveArms to operate on the patient.
Network: is the means of communication between the two consoles.
MasterArms: is the arm through which, the operating surgeon performs surgical maneuvers.
SlaveArms: execute the commands sent from MasterArms on the patient's body directly.

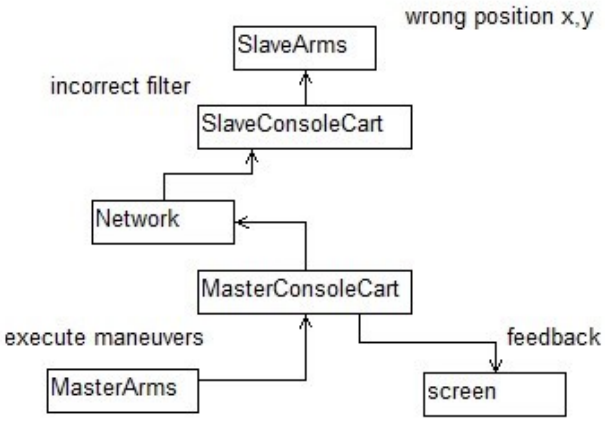

Figure B. 3 high-level control structure for system-level hazards@RemoteSurgery wrong position x,y,z.

c. Identify unsafe control actions

In this phase, we devise a control table using guide words that describe the cause of hazard (Table B.3).

Table B. 3 Unsafe Control Actions for example about @RemoteSurgery wrong position x,y,z.

Control Action	**Unsafe Control Actions** wrong position x,y,z			
	Not providing causes hazard	Providing causes hazard	Too early/too late, wrong order causes hazard	Stopping too soon/applying too long causes hazard
Execute surgical maneuver	#	#	#	surgeon cancels surgical maneuvers due slave console robot arms not responding

Now we address the structure of this hazard using *structure of a hazardous control action*.

Four parts of a *hazardous control action* [Leveson, 2004; 2012].
Source Controller: the controller that can provide the control action
Type: whether the control action was provided or not provided
Control Action: the controller's command that was provided / missing
Context: conditions for the hazard to occur.

Table B.3 { surgeon => Source Controller;
cancels => Type;
surgical maneuvers => Control Action;
due slave console robot arms not responding => Context }

After that we define safety constrains Table B.4

Table B. 4 Safety constrains @RemoteSurgery wrong position x,y,z.

Unsafe control actions	Safety constraints
(Table B.3) surgeon cancels surgical maneuvers due slave console robot arms not responding	surgeon must not perform Maneuver when criteria from SlaveArm are not met.

d. Identify causal factors and control flaws

the operating surgeon performs surgical maneuvers based on what the surgeon sees on the monitor on the master slave. This monitor displays images transmitted from the camera on the slave console. The robot arms in the slave console perform the commands sent from the master console by the operating surgeon. Both consoles are connected through the network. The reason behind the slave console robot arm irresponsiveness is due to Failure in operating software that is responsible for robot arms movements. This indicates that safety constraints must be put in order to monitor the status of the robot arms and availability of feedback for the operating surgeon from the latter to learn the status.

Appendix C STPA-*sec* Process for security

C.1 STPA-sec Process for security

The hazard analysis process in STPA-*sec* [Young and Leveson, 2013; 2014] consists of five phases (Figure C.1).

a. Determining unacceptable losses

From the STPA-*sec* perspective, this phase is executed at the strategic level to determine the context of the system from a comprehensive point of view using a top-down method. In this method, the probabilities and definition of all potential assets losses, which are considered unacceptable for the organisation. The researcher Young [Young, 2014] used "what(s)" and "how" to extract the unacceptable losses, "What(s)-what essential services and functions must be secured against disruptions or what represents an unacceptable loss". After defining services and functions, "How" is used to extract information on how a violation occurs for these functions. "How(s) - that can lead to the undesirable outcomes. The analysis moves from general to specific, from abstract to concrete".
The outcome of the first phase is the definition of vulnerabilities and related loss events.

b. Creating a model of the high level control structure- HLCS

We make graphical specification for the system and its components and the internal components in a high-level control structure way. The HLCS model is built on a lack of constraints. To explain and understand the locations of the main system components and how they are inter-connected with other components and the integration and control points in the system.

c. Identifying unsafe/unsecure control actions

Using the scenario in (Table C.1) and the narrative description in defining high-level unsafe/unsecure control action, and linking that with the outcome of the first phase, vulnerabilities, and the outcome of the second phase which explain the linkage between the components and controlling the receiving and sending flow.
There are four types of potential *unsafe/unsecure* control actions [Young and Leveson, 2013] (Table C.1): *Providing a control action leads to a hazard or exploits the vulnerability; Not providing a control action leads to a hazard or exploits a vulnerability; Providing control actions too late, too early, or in the wrong order leads to a hazard or exploits a vulnerability; Stopping a control action too soon or continuing it too long leads to a hazard or exploits vulnerability.*

d. Developing security requirements and constraints

In this phase, we execute the Refinement of High-Level security Constraints and Requirements process using the scenario in (Table C.2) in order to achieve rigorous constraints for the system.

e. Identifying casual scenarios

We identify causal factors scenarios leading to violation of security constraints using the Control Loop domain model (Figure B.2; Appendix B) and identify how each potentially hazardous/vulnerable States control action identified in Step c" *Identifying Unsafe/Unsecure Control Actions*" could occur.

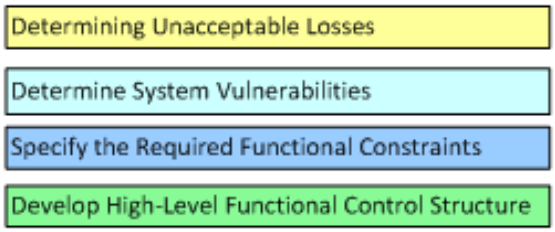

Figure C. 1 System-Theoretic Process Analysis for Security, adapted from [Young, 2014].

Table C. 1 Potentially Unsecure Control Actions, for example about @RemoteSurgery $_{\text{speed sensor}}$**.**

Control Action	**Unsafe/Unsecure Control Actions**			
	Not Providing Causes Hazard	Providing Causes Hazard	Wrong Timing or Order Causes Hazard	Stopped Too Soon or Applied Too Long
# **Speed Sensor**	#	#	#	#

Table C. 2 Defining Security Constraints, for example about @RemoteSurgery $_{\text{speed sensor}}$**.**

Unsafe/Unsecure Control Actions	Security Constraint
# **Speed Sensor**	#

C.2 Running example- STPA-*Sec* Security Corner

We apply the five steps of STPA-*sec* process for security on the example *@RemoteSurgery*, which was addressed from the safety perspective and KAOS-S*a*S in chapter 6, section 6.2. What is addressed in this section will be addressed using STPA-sec. STPA-sec is considered a base scenario [Young and Leveson, 2014; 2013]. Therefore, we run the example based on the above.

a. Determining unacceptable losses

Here, we have to answer the following questions,

system-level accidents (losses)?
Losing connection with the slave console; Losing control on the console; Patient is bleeding.
System-level hazard?
Latency in images transmission.
System-level safety constraints?
Latency value must not exceed an agreed level (value of x ms).

b. Creating a model of the high level control structure (HLCS)

In this phase, we design the high-level control structure for system-level hazards.

Here, we focus the analysis on the latency in transmitting imaged to the operating surgeon. We only presented the main components in the analysis according to STPA.

Components of Figure C.2 are,

MasterConsoleCart: is the platform that transmits the movements the operating surgeon is making according to what is being seen on the master console monitor that displays images transmitted from the slave console on the patient's end.
SlaveConsoleCart: is the platform that received the movements for the robotic arms to operate on the patient.
Camera: is placed inside the patient's body and transmits images to the operating surgeon displayed on the monitor.
Network: is the means of communication between the two consoles.

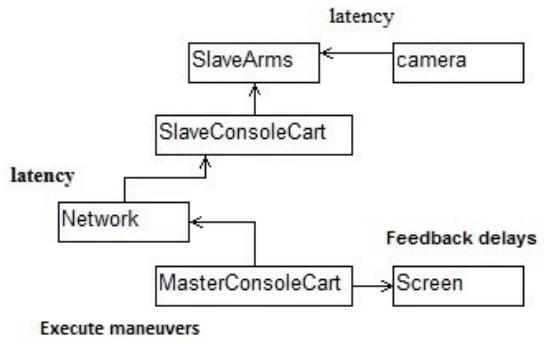

Figure C. 2 high-level control structure for system-level hazards @RemoteSurgery $_{\text{camera latency}}$.

c. Identifying unsafe/unsecure control actions

In this phase, we devise a control table using guide words that describe the cause of hazard (Table C.3).

Table C. 3 Unsafe/Unsecure Control Actions for example about @RemoteSurgery $_{\text{camera latency}}$

Control Action	Unsafe/Unsecure Control Actions $_{\text{camera latency}}$			
	Not providing causes hazard	Providing causes hazard	Too early/too late, wrong order causes hazard	Stopping too soon/applying too long causes hazard
Execute surgical maneuver	#	surgeon performs surgical maneuvers due to image transmission latency	#	#

Now we address the structure of this hazard using *structure of a hazardous control action*.

Four parts of a hazardous control action [Leveson, 2004; 2012].

Source Controller the controller that can provide the control action
Type whether the control action was provided or not provided
Control Action the controller's command that was provided / missing
Context conditions for the hazard to occur.

Table C.3 { surgeon => Source Controller;
performs => Type;
surgical maneuvers => Control Action;
due to image transmission latency => Context }

d. Developing security requirements and constraints

After that we define security/safety constrains Table C.4

Table C. 4 Security/Safety constrains @RemoteSurgery $_{\text{camera latency}}$

Unsafe/Unsecure Control Actions	Security/Safety Constraint
(Table C.3) surgeon performs surgical maneuvers due to image transmission latency	operating surgeon performs surgical maneuvers when image transmission latency is less than X ms.

e. Identifying casual scenarios

The operating surgeon performs surgical maneuvers based on what the surgeon sees on the monitor on the master slave. This monitor displays images transmitted from the camera on the slave console. The robot arms in the slave console perform the commands sent from the master console by the operating surgeon. Both consoles are connected through the network. The reason behind the latency in transmitting images to the operating surgeon's monitor is due to the latency in transmitting data packets above the allowed average value X, which leads to another hazard; unharmonious surgical maneuvers that could lead to an accident in the end.

Licence

Non-exclusive licence to reproduce thesis and make thesis public

I, Mohammed Abulamddi, (date of birth: 09.12.1986),

1. Herewith grant the University of Tartu a free permit (non-exclusive licence) to:

1.1. reproduce, for the purpose of preservation and making available to the public, including for addition to the DSpace digital archives until expiry of the term of validity of the copyright, and
1.2. make available to the public via the web environment of the University of Tartu, including via the DSpace digital archives until expiry of the term of validity of the copyright,

of my thesis

Developing Secure and Safe Systems with Knowledge Acquisition for Automated Specification,

Supervised by
Dr. Raimundas Matulevičius,

2. I am aware of the fact that the author retains these rights.
3. I certify that granting the non-exclusive licence does not infringe the intellectual property rights or rights arising from the Personal Data Protection Act.

Tartu/Tallinn, 25.05.2014