ktu
1922

DONATAS MAŽEIKA

MODEL-BASED SYSTEMS ENGINEERING METHOD FOR CREATING SECURE SYSTEMS

DOCTORAL DISSERTATION

Kaunas
2021

KAUNAS UNIVERSITY OF TECHNOLOGY

DONATAS MAŽEIKA

MODEL-BASED SYSTEMS ENGINEERING METHOD FOR CREATING SECURE SYSTEMS

Doctoral Dissertation
Technological Sciences, Informatics Engineering (T 007)

2021, Kaunas

This doctoral dissertation was prepared at Kaunas University of Technology, Faculty of Informatics, Department of Information Systems, during 2015–2020. The research was funded by the State Studies Foundation.

Scientific supervisors:
Prof. Dr. Lina NEMURAITĖ (Kaunas University of Technology, Technological Sciences, Informatics Engineering – T 007, 2015 - 2016);
Prof. Dr. Rimantas BUTLERIS (Kaunas University of Technology, Technological Sciences, Informatics Engineering – T 007, 2016 - 2020).

Doctoral dissertation has been published in: https://ktu.edu.

Editor:
Brigita Brasienė (Publishing house "Technologija")

KAUNO TECHNOLOGIJOS UNIVERSITETAS

DONATAS MAŽEIKA

MODELIAIS GRĮSTOS SISTEMŲ INŽINERIJOS METODAS SAUGIŲ SISTEMŲ KŪRIMUI

Daktaro disertacija
Technologijos mokslai, Informatikos inžinerija (T 007)

2021 Kaunas

Disertacija rengta 2015–2020 metais Kauno technologijos universitete, Informatikos fakultete, Informacijos sistemų katedroje. Mokslinius tyrimus rėmė Valstybinis studijų fondas.

Moksliniai vadovai:
Prof. dr. Lina NEMURAITĖ (Kauno technologijos universitetas, technologijos mokslai, informatikos inžinerija – T 007, 2015 m. – 2016 m.);
Prof. dr. Rimantas BUTLERIS (Kauno technologijos universitetas, technologijos mokslai, informatikos inžinerija – T 007, 2016 m. – 2020 m.).

Interneto svetainės, kurioje skelbiama disertacija, adresas: http://ktu.edu

Redagavo:
Brigita Brasienė (leidykla „Technologija”)

ACKNOWLEDGEMENT

The author of the dissertation would like to express sincere gratitude to the initial scientific supervisor Prof. Dr. Lina Nemuraitė for introducing to research work and assisting in various situations. Moreover, the author would like to acknowledge Prof. Dr. Rimantas Butleris who took the role of the thesis supervisor and led it until the final stages. The author of the dissertation highly appreciates his support and the given freedom in writing this dissertation.

The author wishes to express deepest gratitude to his wife Eglė for her endless support and belief that this dissertation will be finished. The author would like to thank his mother Irma and father Kazys-Romualdas for always encouraging to choose any path in life as long as it feels the right one.

The author would like to extend thanks to "No Magic Europe" (now a part of "Dassault Systèmes") for their comprehensive and professional support. The author would like to thank the whole Department of Information Systems of Kaunas University of Technology for their knowledge and help during the PhD studies.

Finally, the author's honest thanks go to all magnificent friends and colleagues.

TABLE OF CONTENTS

TERMS AND ABBREVIATIONS

Term	Description
CAD	Computer-Aided Design.
CAE	Computer-Aided Engineering.
CASE tool	Computer-Aided Software Engineering tool. It is a computer program that provides automated assistance for software and system development.
CHASSIS	Combined Harm Assessment of Safety and Security for Information Systems.
CIA	Confidentiality, Integrity and Availability. Also known as the CIA triad or CIA model.
CPS	Cyber-Physical System. It is a physical system whose operations are monitored, controlled, coordinated and integrated by computer-based algorithms.
DSL	Domain Specific Language. It is a computer language developed to a particular domain. This is in contrast to a general-purpose language, which is broadly applicable across domains.
DSML definition framework	Domain-Specific Modelling Environment based on UML profiles.
ECU	Electronic Control Unit. It is an embedded system in automotive electronics that controls one or more of the electrical systems in a vehicle.
fUML	Foundational UML. It is a subset of the standard Unified Modelling Language (UML) for which there are standard, precise execution semantics.
HSUV	Hybrid Sport Utility Vehicle.
INCOSE	International Council On Systems Engineering.
ISMS	Information Security Management System. It is a framework of policies and controls that manage security and risks systematically and across the entire enterprise information security.
IETF	Internet Engineering Task Force. It is an open standards organization, which develops and promotes voluntary Internet standards.
MBSE	Model-Based Systems Engineering.
MBSEsec	The name of the method which is introduced in this thesis. As well named as the *Model-Based Systems Engineering method for creating secure systems* or the *MBSE security method* in the dissertation.
MDA	Model-Driven Architecture.

Modelling Language	Any artificial language that can be used to express information or knowledge in a structure that is defined by a consistent set of rules.
MOF	Meta-Object Facility. The Object Management Group (OMG) standard for model-driven engineering.
OCL	Object Constraint Language. It is a declarative language describing rules applying to UML models.
OCTAVE	Operationally Critical Threat, Asset, and Vulnerability Evaluation. A framework for identifying and managing information security risks.
OMG	Object Management Group, the consortium aimed at setting the wide range of technology standards for distributed object-oriented systems and model-driven development.
OOSEM	Object-Oriented Systems Engineering Methodology.
PDCA Model	Plan–Do–Check–Act. It is an iterative, four-stage model for continually improving processes, products or services.
Profile	Profile is a lightweight extension mechanism to the UML language.
SE	Systems Engineering (SE).
SysML	Systems Modelling Language. It is the OMG standardized general-purpose graphical modelling language for specifying, analysing, designing and verifying complex systems that may include hardware, software, information, personnel, procedures and facilities.
SYSMOD	Weilkiens Systems Modelling Process.
UML	Unified Modelling Language. It is the OMG standardized general-purpose modelling language used in a very broad scope that covers a large and diverse set of application domains, including the field of software engineering and object-oriented software-intensive systems.
V-Model	V-model is one of the most common graphical representation of a systems engineering lifecycle.

FIGURES

TABLES

1. INTRODUCTION

1.1. Motivation[1]

Modern systems among industries such as automotive, medical devices, aerospace and defence are becoming extremely complex; therefore, traditional engineering methods are not enough for their successful realization. The systems have become more complex due to many factors, to name a few:

- Increased spectrum of technologies: complex systems have become cyber-physical systems (CPS) and now depend upon the seamless integration of computational algorithms and various physical components [1];
- Increased customer demands for more sophisticated systems and market or military competition [2];
- Systems consist of many components interacting in a network structure, and usually, these components are physically and functionally heterogeneous [3].

The discipline of systems engineering (SE) was initiated and developed to manage and unite work results of multidisciplinary engineering teams. The goal of SE is a successful realization of systems with the focus on gathering customer needs and defining required functionality early in the development cycle as well as documenting requirements, then proceeding with design synthesis and system validation [4]. Nowadays, organizations that cannot cope with systems complexity have switched (or are switching) from a document-based approach to a model-based approach in the SE activities. International Council on Systems Engineering (INCOSE) emphasizes MBSE importance, and they envision that MBSE will become a synonym of SE by 2025 [5]. The advantages of using models instead of documents in SE include the following [6, 7, 8]:

- Increased systems engineering efficiency by:
  - reusing existing projects or common components to support design and technology evolution;
  - enabling impact analysis of requirements changes;
  - improving communication across a multidisciplinary team;
  - enabling auto-generation of documentation.
- Reduced risk by early and iterative requirements validation and design verification;
- Managed complexity.

There are a few methods that guide users on how to get all the MBSE benefits when creating a system design model; sadly, almost all the analysed methods do not include the security analysis at the early stage of system design. Conversely, there are several tools and approaches that allow performing security analysis at the initial phase of systems creation (e.g., Misuse Cases, Abuse Cases, Secure-Tropos, CHASSIS); however, they are disjointed from the systems engineering [9, 10].

[1] The material in the "Motivation" section was presented by Mažeika et al. in [88, 94].

Many researchers in their studies [11, 12, 13, 14] agree that there is a need to identify and tackle security risks during the systems engineering lifecycle. Nguyen et al. state that security objectives (such as confidentiality, integrity and availability) should be considered together with the business logic very early, which is crucial in engineering secure systems. Thus, MBSE could be a key helper because of the opportunity to manipulate models on a higher abstraction level, possibility to tailor generic modelling language (e.g., UML and SysML) with the security-related concepts and performing reasoning with external analysis tools [12]. Nowadays, the MBSE activity mostly focuses on the design phase, which is usually done by the systems engineers. When developing complex systems, the security analysis is often conducted in parallel with the design phase. Papke argues that security engineers and systems engineers should work together, and a joint design process or framework is needed in order to define security aspects in a common model [13].

The authors recognize that the biggest value of MBSE activities is gained when system validation and verification are performed at the early phase of system design, especially in terms of change of cost [15, 16]. In such case, the defects could be fixed with less impact and the rework prevented in the later phases, thus mitigating uncertainties to cost and schedule [16]. The same principles apply in the security field: the risk identification and mitigation are the most effective and maximize the return on investment if it is integrated into the design process and utilized in the early stages [17].

1.2. Object and scope of the research

The research object of this work is the MBSE method for creating secure complex systems formalized with the UML language.

The scope of the research encompasses the following fields:

- Systems Engineering (SE) and Model-Based Systems Engineering (MBSE),
- Security Requirements Engineering,
- Modelling approaches and techniques for security analysis,
- Security Risk Management,
- Standards and methods for creating domain specific language and formalizing the MBSE security method.

Figure 1.1. emphasizes that the proposed MBSE method for creating a secure system is a part of system problem domain definition, which covers activities that are performed at the early stage of the system design lifecycle (e.g., capturing stakeholder needs, defining system context, modelling functional analysis, creating a logical design, defining system parameters and specifying system requirements). The problem domain definition is one of the main viewpoints for SE. Traditionally, in MBSE methodologies and Enterprise Architecture Frameworks, there are two viewpoints to manage the abstraction complexity: one to define a problem in order to understand it, the other to provide one or multiple alternative solutions to solve it [18].

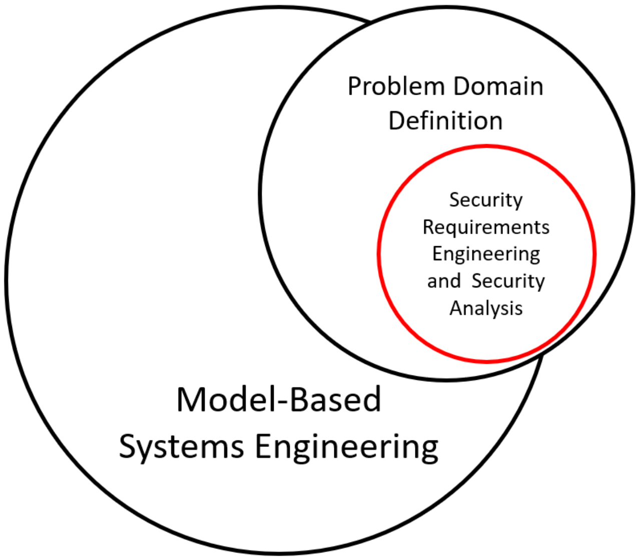

Problem Domain Definition
Security Requirements Engineering and Security Analysis
Model-Based Systems Engineering

Figure 1.1. The relationship between MBSE and research object

1.3. Problem statement and research questions

One of the most important challenges that organizations are trying to solve while creating a new system is how to develop a secure system. Traditionally, the system is treated as a secure system if the principles of Confidentiality, Integrity and Availability are guaranteed.

Nowadays, the system security engineering field includes a variety of methods and techniques for tackling security risks; however, they are disjointed from each other as well as from SE. As MBSE serves as an umbrella for connecting various disciplines, this disparity between security and SE becomes more evident. The problem of this dissertation focuses on the lack of MBSE methods for tackling security issues at the early stage of system creation.

This dissertation should give answers to the following research questions:

1. Is MBSE a suitable application for defining and managing security requirements and conducting security analysis for complex cyber-physical and software systems at the early stage of system creation?
2. Are the UML Profiles and MOF standard the right techniques and standards for creating and formalizing the domain-specific language and MBSE security method?
3. How can security requirement engineering and security analysis activities be included in the MBSE process to design a secure system and leverage MBSE advantages?
4. What are the security concepts that should be introduced in systems modelling language in order to support security aspects during the early stages of system development?
5. What domain specific extensions (e.g., stereotypes, diagrams, verification rules) are needed for security analysis?

6. Can the automated MBSE tools, including but not limited to simulation, verification and validation, change impact analysis, single source of truth, be successfully applied in the security field by using the proposed method?
7. Does the proposed MBSE security method allow completely, concisely, correctly and consistently model security aspects of both cyber-physical and software systems in the CASE tool?

1.4. Aim and objectives

The main aim of this research is to find an effective way to solve the secure system creation challenge at the early stage of system development by taking advantage of Model-Based Systems Engineering.

Research tasks:

1. To analyse research literature, methods, applications and tools related to:
   1.1. Systems Engineering (SE) and Model-Based Systems Engineering (MBSE),
   1.2. Security Requirements Engineering,
   1.3. Modelling approaches and techniques for security analysis,
   1.4. Security Risk Management,
   1.5. Standards and methods for creating domain specific language and formalizing the MBSE security method.
2. To develop a formalized MBSE method for creating secure complex systems.
3. To perform an experiment for evaluating the suitability of the created method and evaluate the research results.

1.5. Research methodology

The research methodology followed in this thesis is based on a traditional design science research pattern [19]. The starting point was the evaluation of the state-of-the-art of the existing literature in SE, MBSE, security requirements engineering and security risk management fields. This initial evaluation of the state-of-the-art literature analysis aimed to identify the limitations and potential needs in SE, MBSE and security areas, align concepts and techniques and select the core elements for the domain specific language and the MBSE security method.

Next, the feasibility survey was conducted in order to validate the business needs before creating the MBSE security method.

The next chapter of “MBSE method for creating secure systems” started by presenting the standards and tools that are needed to define domain-specific language and formalize the MBSE security method (i.e., UML 2.5 Profiling capability, MOF standard, DSML definition framework). Next, a classical modelling language design approach where the key concepts of the domain should be determined at first and then a new language could be created to support it was used [20]. The security concepts that were identified in the literature analysis part were mapped and represented in the domain model; then, UML profile was prepared according to the domain model. The requirements of MBSE security method implementation (which as well serves as guidelines) were defined in textual form by following IETF RFC2119 recommendations.

The evaluation part consisted of several iterations. Firstly, the qualitative evaluation of the proposed MBSE security method was done by surveying experts from the MBSE, engineering and academic fields. Then, two case studies were modelled using the suggested MBSE security method in which the viability for cyber-physical and software systems were presented. Finally, these case studies were experimentally tested against four criteria: completeness, correctness, conciseness and consistency.

1.6. Defended statements

The statements that were defended by the research are as follows:

1. MBSE is a suitable application for defining and managing security requirements and conducting security analysis for complex cyber-physical and software systems at the early stage of system creation.
2. The UML 2.5 Profiling capability and MOF standard are the right techniques for creating and formalizing the domain-specific language and MBSE security method.
3. The automated MBSE tools, including but not limited to simulation, verification and validation, change impact analysis, single source of truth, can be successfully applied in the security field by using the proposed method.
4. All the artefacts that are mandatory for defining security-related documentation (i.e., comparing with the ISO/IEC 27001:2013 standard) can be correctly, concisely and consistently modelled in a model-based environment with a suggested MBSE method for both cyber-physical and software systems.

1.7. Major contributions and novelty

The scientific novelty and major contributions of this thesis are listed below:

1. The thesis introduces the security domain model that maps concepts and techniques from the modelling approaches for security analysis and security requirement engineering. The mapping and the security domain model help security and system engineers to understand and compare wide range security terms and techniques that could be used at the early stage of system design.
2. It introduces a novel MBSE method for creating secure systems. It allows specifying and analysing security aspect together with the system model for complex systems. The suggested MBSE method covers the full spectra of security phases, starting with security requirements, continuing on assets, model threats and risks and finishing with control objectives and controls. The use of model-based techniques ensures that the security and system artefacts are aligned at the early phase of system design, and MBSE benefits are extended to security engineer discipline.
3. The author's suggested security method is one of the first methods in the MBSE field at the time of publication.

1.8. Practical significance

The key practical significance of this research is the step towards linking systems engineering and security engineering disciplines via model-based environment. The expected practical results of the research:

- The MBSE method for creating secure systems is prepared and can be used for designing any complex system with an MBSE CASE tool.
- The MBSEsec profile and DSML definition package was prepared with the MagicDraw 19.0 CASE tool and can be installed as a plugin in any compatible tool. Moreover, the requirements of the MBSE method implementation unambiguously define how this method can be recreated with any tool.
- The MBSEsec method provides guidelines on how to use and take leverage of the MBSE benefits (e.g., model verification and simulation, change impact analysis, traceability).
- The proposed MBSEsec method is aligned with the ISO/IEC 27001:2013 security standard which is used by many engineering organizations.
- The thesis presents two case studies for automotive and software system domains.
- The practical significance was validated in two surveys: feasibility and qualitative expert evaluation.

1.9. Scientific approval

Two articles presenting dissertation results were published in peer-reviewed scientific journals that are indexed in the Clarivate Analytics Web of Science (CA WoS) database. Moreover, the results of this research were presented in three international conferences in Norway, Australia and Hungary and in one international workshop in Lithuania. The corresponding publications were published in the conference proceedings. A detailed list of publications is provided in Chapter 7 “List of publications of Donatas Mažeika on the theme of dissertation”.

1.10. Structure of the dissertation

The dissertation consists of an introduction of the thesis, four main chapters, general conclusions, references, a list of the author’s publications and appendixes. Moreover, the terms and abbreviations, lists of figures and tables are presented at the beginning of this work. The total scope of the thesis is 99 pages; it includes 48 figures and 12 tables.

Please note that figures and tables without citations are created by the author of the dissertation.

2. ANALYSIS OF RELATED WORKS

Many researchers from all over the world and from Lithuania have been exploring various aspects of SE and MBSE. Several leading research works were developed or are under development in this field in the Department of Information Systems (ISK) of Kaunas University of Technology in partnership with “Dassault Systèmes” (previously known as “No Magic”) [21]. Even though the MBSE field is widely researched and used in practice, there are not many attempts to standardize how the security analysis can be conducted in a model-based environment within the system engineering process.

Looking from a different perspective, the field of information and cyber security is widely researched for the software engineering discipline. There are several industry-acceptable methods to ensure secure software development throughout all phases of the development process, including the waterfall-based Microsoft Security Development Lifecycle (SDL) method [22] and the NIST framework for Security Considerations [17] as well as the Microsoft Security Development Lifecycle for Agile Development [23]. Herewith, the notable research works were conducted by Lithuanian researchers with a special focus on cryptography, application layer protocols and network communications. Kilčiauskas et al. presented the research of authenticated key agreement protocol based on provable secure cryptographic functions for e-banking systems [24]; Kajackas et al. assessed cyber-attacks influence over an internet network [25]. Janulevičius defended a doctoral dissertation on the security threat categorization taxonomy for virtualized systems [26].

The following subsections present state-of-the-art analysis of related works that reveal the general background information on the systems engineering research field. This covers SE, MBSE, Systems Modelling Language (SysML) and the comparison of leading MBSE methodologies. Next, the security requirements engineering domain and the modelling approaches for security analysis are analysed. There, the key security concepts and techniques are selected and aligned. Next, the security risk management application is overviewed, and the ISO/IEC 27001:2013 information security standard is presented in greater detail.

2.1. Systems Engineering

The Systems Engineering (SE) discipline promotes a holistic approach to design, analysis and management of complex engineering projects (e.g., in automotive, aerospace, defense industries). SE started emerging in the 1990s as a preferred approach for enabling engineers to cope with the complexity and manage system projects that satisfy stakeholders' needs while limiting costs, development time and other resources [4, 27].

When talking about system engineering, it is worth defining the system, as it is the main subject of this discipline. A complex system can be generally defined as a set of interrelated components that interact with each other to accomplish a desired goal. It may combine hardware and software components, include processes, data and humans. Organizations and researchers that are involved in the SE research field give the following thoughts related to the system:

- The International Council on Systems Engineering (INCOSE) defines a system as “*a construct or collection of different elements that together produce results not obtainable by the elements alone. The elements, or parts, can include people, hardware, software, facilities, policies, and documents; that is, all things required to produce systems-level results*” [4].
- IEEE Reliability Society has derived the following definition: “*a system is a group of interacting elements (or subsystems) having an internal structure which links them into a unified whole. The boundary of a system is to be defined, as well as the nature of the internal structure linking its elements (physical, logical, etc.). Its essential properties are autonomy, coherence, permanence, and organization*” [28].
- The ISO/IEC 15288:2015 systems engineering standard states that system is: “*an integrated composite that consists of one or more of the processes, hardware, software, facilities, and people that provides a capability to satisfy a stated need or* objective” [29].

One of the most known and industry-approved definitions of the SE discipline is provided by INCOSE: “*Systems Engineering is an interdisciplinary approach and means to enable the realization of successful systems. It is focused on defining customer needs and required functionality early in the development cycle, documenting requirements, and then proceeding with design synthesis and system validation while considering the complete problem: operations, cost and schedule, performance, training and support, test, manufacturing, and disposal*” [4]. SE is a multidisciplinary approach, and it combines and integrates multiple disciplines, as illustrated in Figure 2.1. The central role for the SE discipline is a systems engineer who usually oversees the whole perspective of the system; gathers, clarifies and specifies requirements; performs the parametric analysis and trade-offs; checks if different components can be integrated (interfaces compatibility) and takes early action to avoid defects [4].

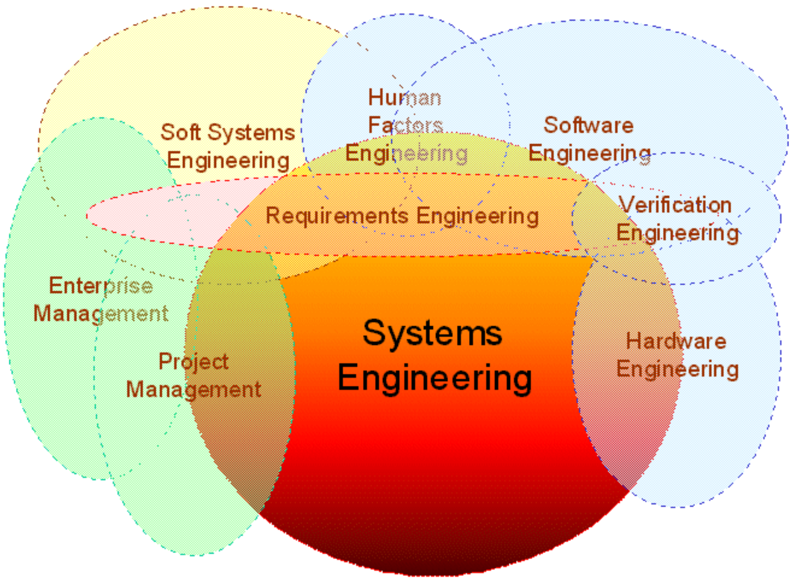


Figure 2.1. Disciplines related to Systems Engineering [30]

The SE lifecycle and its main steps are often graphically represented using the V-Model [31, 32]. The comprehensive example of V-Model by [32] is presented in Figure 2.2. There are several different versions of the V-Model; however, most of them share core similarities, which can be summarized as follows:

- The SE engineering process starts with a feasibility study and concept exploration. Usually, this step includes gathering the needs of the stakeholders in order to confirm what are the goals, objectives and key requirements of the system under design.
- Next, the stakeholder needs are refined and converted into functional and non-functional system requirements.
- In the next steps, the system logical design is prepared according to the requirements from the previous step. Most often, the high-level logical design is created at first, and the more detailed solution design is created afterwards.
- In the last step of the left-hand side, the implementation details for software and hardware components are specified.
- The right-hand side of the V-model is dedicated to testing, integration, validation and verification of the components, subsystems and systems that are defined in each stage on the left-hand side of the V-model.

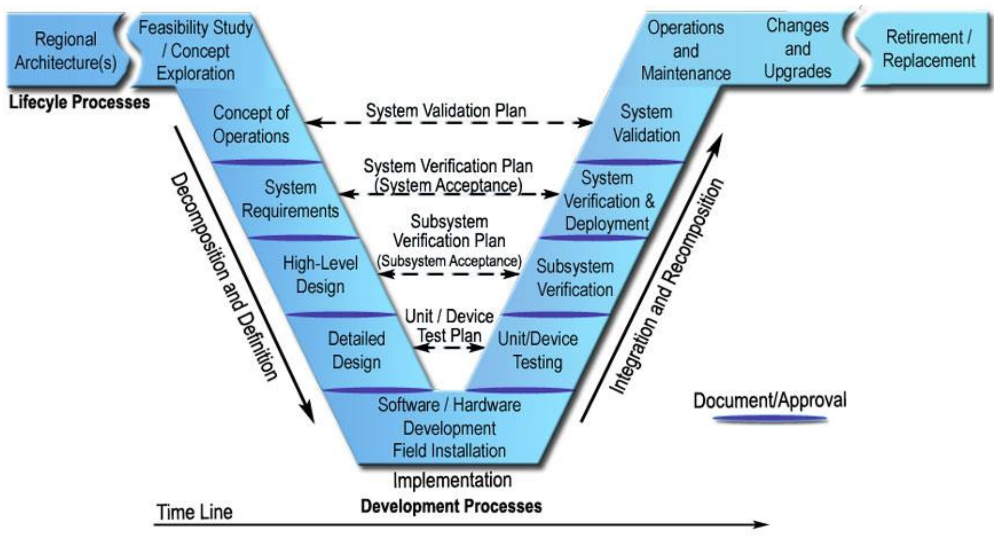


Figure 2.2. V-model representing systems engineering lifecycle [32]

Initially, SE relied mostly on document-based artefacts, i.e., requirements documents, system specifications, schematic block diagrams, interface control documentation, system architecture descriptions. This information is frequently maintained by different persons and captured in many different files and formats, including text, non-standard schemes and spreadsheets. As a result, the document-based approach may lack precision: there could be inconsistencies from one artefact to another. Moreover, it leads to the difficulties of maintaining and reusing the information [33]. All these challenges are trying to be solved by Model-Based Systems Engineering, which is analysed in the next section.

2.2. Model-Based Systems Engineering

The Model-Based Systems Engineering (MBSE) methodology leverages models for the whole spectra of system engineering activities (i.e., requirements, logical architecture, system behaviour, integrations, validation and verification) and makes the model a central figure. The change from document-based to model-based approach in SE can be compared with the paradigm shift that happened in engineering and industrial design industries with Computer-aided engineering (CAE) and Computer-aided design (CAD) software [5]. Moreover, this transformation enables organizations to move from waterfall/linear SE approach to more agile methods [34].

INCOSE defines MBSE as *"the formalized application of modeling to support system requirements, design, analysis, verification and validation activities beginning in the conceptual design phase and continuing throughout development and later life cycle phases"* [35]. Both SE and MBSE definitions agree that systems engineering should support requirements, design, analysis and validation activities; however, the MBSE definition emphasizes that this support is realized by modelling.

The MBSE methodology promises to ease systems engineers' challenges in communication across different engineering disciplines, especially in terms of completeness and consistency [36]. In order to solve these challenges and successfully adapt the MBSE, three aspects should be mastered: common SE language, method and CASE tool [18].

Language. MBSE was accelerated with the effective usage of Unified Modelling Language (UML) [37] and the practice of Model-Driven Architecture (MDA) [38]. Prior to this, there were many attempts to apply the UML language for SE; however, these tries were not successful [39] because of complexity of language, and it was non-natural for solving SE domain-specific problems [40]. Due to these reasons, the OMG group, in partnership with INCOSE, started working on domain-specific language creation in 2001, and the first version of the Systems Modelling Language (SysML) was released in 2006 [41].

SysML is based on UML 2.0, and it provides a focused set of UML diagrams and presents several new or modified diagrams for modelling complex systems that include software, hardware, procedures, data and other system components [34]. All the SysML diagrams are presented in Figure 2.3. As SysML is a profile of UML, it can be integrated with other OMG UML-based standards, such as Unified Architecture Framework (UAF), Object Constraint Language (OCL), executable UML models (fUML), etc.

One remark about SysML and UML should be noted: these languages are visual modelling languages that are not depended to any SE methodology [34].

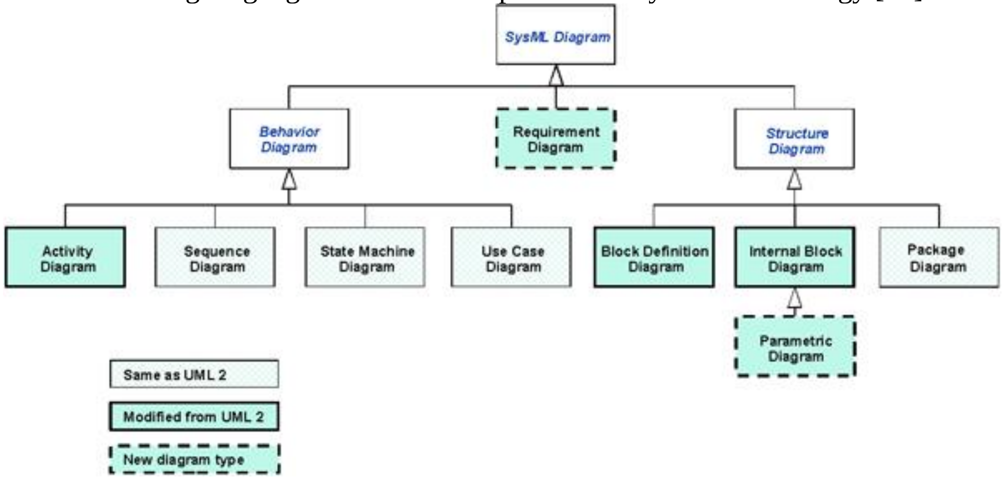


Figure 2.3. SysML diagram taxonomy [42]

CASE tool. All the MBSE benefits are achievable only with the proper modelling tool. Firstly, the MBSE should be carried out with a modelling tool in which model elements are created underneath and represented in different views, not with diagramming/drawing tool. Secondly, the tool should support the key MBSE capabilities, such as one source of truth, automated document generation, simulation, model reuse, change impact analysis, consistency and completeness check. There are many CASE tools for SE with their own strengths and weaknesses. This dissertation is carried out using the MagicDraw toolset. The benefits of using this tool were presented in several researches [43, 36, 44].

Method. System modelling language (i.e., SysML) is not sufficient to run the MBSE project successfully: it provides only semantics of the language but not how to use it practically and methodically [39]. The language must be combined with a method or methodology and conducted with the proper CASE tool in order to complete the MBSE project. The next section presents the analysis of MBSE methodologies.

2.3. Model-Based System Engineering methodologies

This section provides the analysis and comparison of the leading MBSE methodologies. The MBSE methodology should be understood as the set of processes, methods and tools used to support the discipline of SE in a model-based way [34].

The starting point for selecting key MBSE methodologies used among systems engineering practitioners is the INCOSE repository of MBSE methodologies [45] and comprehensive survey of MBSE methodologies conducted by Estefan [34]. Moreover, a new approach of MagicGrid is included in this analysis, as it is a synthesis of widely known MBSE methodologies, and it is successfully applied in real-world projects [46].

The six MBSE methodologies are looked over in this section:

1. Object-Oriented Systems Engineering Methodology (OOSEM),
2. IBM Harmony methodology for SE,
3. Weilkiens Systems Modelling Process (SYSMOD),
4. NASA JPL State Analysis,
5. Vitech MBSE Methodology,
6. MagicGrid (also known as MBSE Grid).

A review of each methodology presents the main and additional capabilities, looks if it is suitable for the early phase of system development, checks if it supports security analysis, highlights gaps, etc.

2.3.1. Object-Oriented Systems Engineering Methodology

The Object-Oriented Systems Engineering Methodology (OOSEM) was originally created by system engineering practitioners from Lockheed Martin and the Systems and Software consortium [47]. OOSEM combines object-oriented software engineering principles, a model-based system design approach and traditional/waterfall-style system engineering practices. OOSEM is based on UML and SysML languages.

This methodology suggests four main activities for system engineers:

1. Analyse and capture stakeholder needs and system requirements;
2. Model logical system design;
3. Define allocated architecture;
4. Validate and verify the system.

Additionally, the methodology provides guidelines for optimizing and evaluating system architecture.

The OOSEM methodology is in compliance with standard ISO/IEC-15288, which is dedicated to aligning the procedures used by any organization or project throughout the full lifecycle of a system [48]. System Engineering processes in the ISO-152888 standard are organised into five categories: Agreement, Enterprise, Project, Technical and Special. The steps from the ISO-15288 standard helps to identify the sequence of the processes needed to deliver the essential products of the system development.

2.3.2. IBM Harmony methodology for SE

The IBM Harmony methodology for SE (Harmony-SE) is based on the rational unified process (RUP) and uses the SysML diagrams. The suggested SE activities follow the classical system engineering “V” diagram. The left side of the “V” describes the top-down design flow, while the right-hand side shows the bottom-up integration phases from unit test to the final system acceptance [49].

The Harmony-SE workflow has three main phases:

1. Requirements analysis,
2. System functional analysis,
3. Design synthesis.

The MBSE techniques that support the requirements analysis phase are SysML Requirements diagram and Use Cases model. In the system functional analysis phase, each use case is transformed into an executable model, and the related system requirements are verified using model execution. The main executable models in the design synthesis phase are Architectural Analysis Model and System Architecture Model.

The Harmony-SE methodology has the approach on how the MBSE model can be integrated with the Software Implementation model. The principal diagram of such integration is presented in Figure 2.4.

In addition, Harmony-SE supports dependability analysis as a parallel activity to requirements and system design. This is an optional activity (e.g., the system under design is not needed to be high-reliability, safety-critical or security-sensitive system), and it should ensure that the system under design meets the security, reliability and safety needs of the stakeholder. There are the IBM Rhapsody profiles for Harmony-SE for defining dependability domain; however, its capabilities may be limited if comparing to specialized tools [50]. As it is presented in [51], the UML Security Analysis profile has a Security Analysis diagram, which is like a Fault Tree Analysis but for security rather than safety. It allows defining dependencies between assets, vulnerabilities, attacks and security violations.

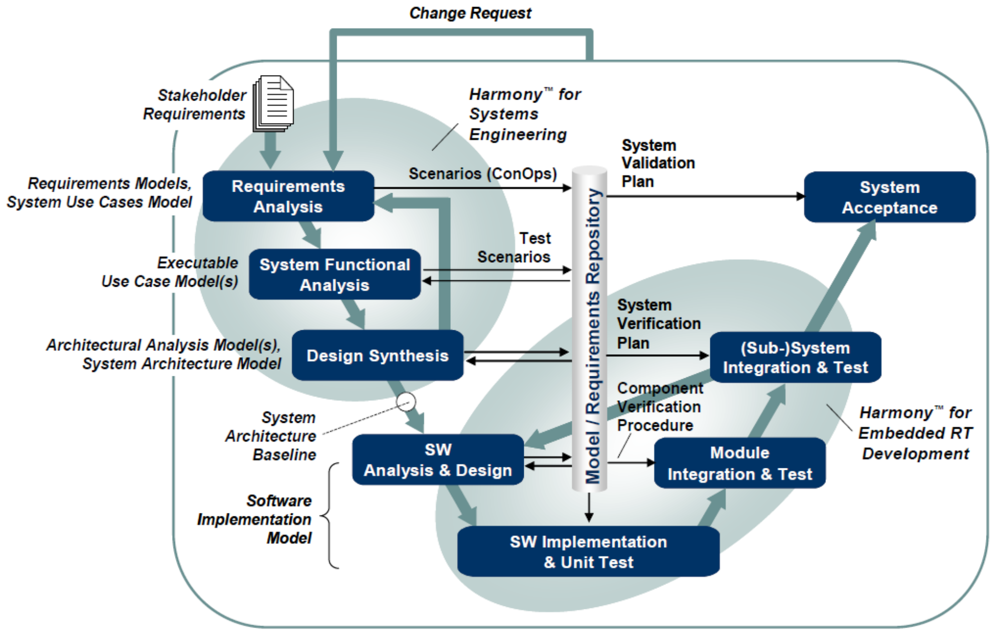

Change Request
Stakeholder Requirements
Harmony™ for Systems Engineering
System Validation Plan
Requirements Models, System Use Cases Model
Requirements Analysis
Scenarios (ConOps)
System Acceptance
Executable Use Case Model(s)
System Functional Analysis
Test Scenarios
Model / Requirements Repository
System Verification Plan
Architectural Analysis Model(s), System Architecture Model
Design Synthesis
(Sub-)System Integration & Test
Component Verification Procedure
System Architecture Baseline
SW Analysis & Design
Module Integration & Test
Harmony™ for Embedded RT Development
Software Implementation Model
SW Implementation & Unit Test

Figure 2.4. Harmony for SE and Harmony for Embedded Systems [49]

2.3.3. Weilkiens Systems Modelling Process

The Weilkiens Systems Modelling Process (SYSMOD) provides process definition, guidelines and examples on how to define requirements and system architecture with the SysML language [52]. The main methodology activities are:

1. Identify stakeholders;
2. Elicit requirements;
3. Define the system context;
4. Analyse requirements, e.g., with the Use Case diagram;
5. Define domain model;
6. Define the system architecture on different levels (functional, logical, physical).

Starting with the description of the system context, the needs of the system are captured and modelled. The use case specification allows clarifying requirements and working scenarios. The processes of system are created in parallel. Finally, the internal structure of the system is created, parameters are defined, and behaviour is modelled.

SYSMOD provides guidelines for additional activities too, e.g., for functional architectures or variant modelling.

2.3.4. NASA JPL State Analysis

The JPL state analysis methodology was developed by the California Institute of Technology Jet Propulsion Laboratory (JPL). It is a formal methodology, which is based on a state control architecture where state is defined to be "*a representation of* *the momentary condition of an evolving system*," and models describe how state evolves [53].

The methodology provides three main activities for state modelling:

1. Modelling behaviour according to the state variables and relationships between them;
2. Designing state-based software (methods to achieve objectives);
3. Engineering goal-directed operations (preparing detailed scenarios for mission objectives).

State and models provide instruments for designing a system, predicting a future state, controlling it towards a desired state and formally assessing performance. State analysis should be managed in the State Analysis SQL database [34]. Alternatively, the state analysis methodology can be conducted with the SysML extension on the MBSE modelling tool [54].

2.3.5. Vitech MBSE Methodology

The Vitech MBSE Methodology uses System Definition Language (SDL), and it allows modelling the syntax (structure) and semantics (meaning) of a complex system. The system model can be specified in the form of schema or ontology. The SDL language inherits many diagrams and constructs from the traditional SE artefacts, e.g., Enhanced Function Flow Block Diagrams (EFFBDs), N2 charts, State-transition diagrams [55].

The main domains of Vitech methodology are listed below:

1. Process Domain (SE activities),
2. Source Requirements Domain,
3. Behaviour Domain,
4. V&V Domain,
5. Architecture Domain.

The methodology suggests using a linear strict process known as "Onion Model" to work incrementally and increase levels of detail during the system specification process. The engineers must fully complete a layer and only then move to the next one.

2.3.6. MagicGrid

The MagicGrid approach (also known as MBSE Grid) was originally proposed in [46], where the process for problem definition layer, which is essential at the early phase of system development, was described. The follow-up article presented guidelines for the solution layer; moreover, it introduced traceability among views [18].

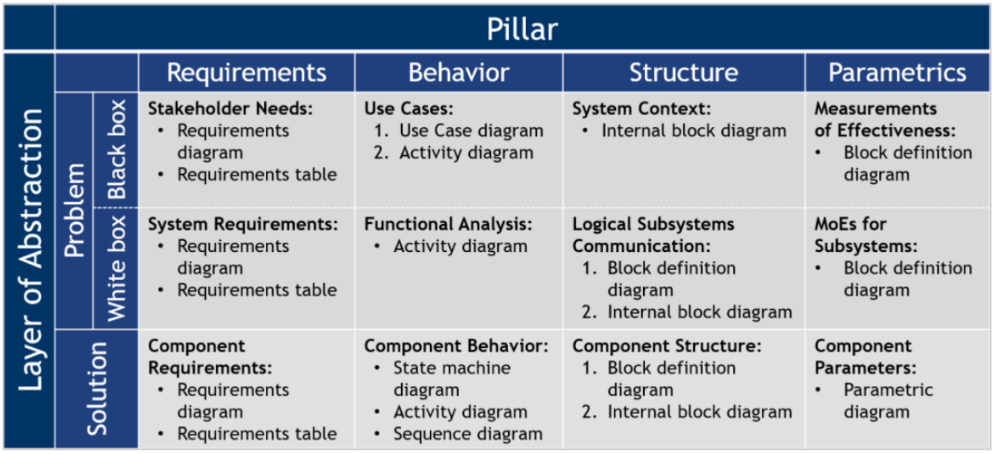
The MagicGrid approach is influenced by the MBSE adoption projects from the transportation and defence industries. The approach is based on the framework organized in a matrix view (see Figure 2.5.), and it defines the modelling process, reveals what model artefacts should be produced in each step of system specification and design and explains how to manage traceability relationships (both horizontal and vertical).

The approach guides how to specify four main aspects of systems engineering in different layers of abstraction:

1. Requirements (stakeholder needs, system and component requirements);
2. Behaviour (use cases, functional analysis, component behaviour);
3. Structure (system context, logical subsystems communication, component behaviour);
4. Parametrics (measurements of effectiveness, component parameters).

The layers of the abstraction are grouped into two viewpoints, i.e., Problem and Solution. The problem domain is as well divided into two sub-viewpoints, i.e., Black Box and Whitebox definitions.

MagicGrid is fully compatible with the SysML language. It is oriented to the creation of a system model; however, there are plans to extend it and include support of system variants, engineering analysis and verification & validation.



			Pillar			
Layer of Abstraction			Requirements	Behavior	Structure	Parametrics
	Problem	Black box	**Stakeholder Needs:** • Requirements diagram • Requirements table	**Use Cases:** 1. Use Case diagram 2. Activity diagram	**System Context:** • Internal block diagram	**Measurements of Effectiveness:** • Block definition diagram
		White box	**System Requirements:** • Requirements diagram • Requirements table	**Functional Analysis:** • Activity diagram	**Logical Subsystems Communication:** 1. Block definition diagram 2. Internal block diagram	**MoEs for Subsystems:** • Block definition diagram
	Solution		**Component Requirements:** • Requirements diagram • Requirements table	**Component Behavior:** • State machine diagram • Activity diagram • Sequence diagram	**Component Structure:** 1. Block definition diagram 2. Internal block diagram	**Component Parameters:** • Parametric diagram

Figure 2.5. MagicGrid mapping to SysML [46]

2.3.7. Comparison of MBSE methodologies

This section provides the comparison and summary of the analysed MBSE methodologies. All the analysed methodologies were evaluated against seven criteria (see Table 2.1.).

The first criterion checks if MBSE methodology suggests activities for an early phase of the system development lifecycle (e.g., defining stakeholder needs, system requirements, use cases, functional analysis). All the methodologies, except JPL State Analysis, offer such capability. The JPL State Analysis methodology originally was dedicated to formally define complex control systems, leaving aside requirements and other early-phase activities.

The second item looks at what additional activities can be done with the selected MBSE methodology. Vitech MBSE and MagicGrid suggest activities only for SE in a current version. JPL State Analysis provides tools for the formal system validation and verification. Other methodologies support activities that are closely related to SE, such as variant modelling, software implementation model, architecture optimization.

The third item presents the base modelling language. Most of the compared methodologies are based on SysML/UML, which is a de facto language for SE [34].

JPL State Analysis uses the SQL language, which is designed to manage data held in a relational database management system. Moreover, JPL State analysis has a mapping with SysML concepts [54]. The Vitech MBSE methodology is based on the proprietary SDL language.

The fourth and fifth criteria look if the modelling methodology supports iterative/agile cycles and provides validation/verification tools. This can help mitigate defects early in the system design lifecycle by testing and validating system models frequently and quickly getting feedback and needed decisions. All the methodologies, except SYSMOD, have validation and verification activities; 4 of 6 methodologies support the iterative process.

The sixth question asked if the security analysis is supported by the analysed MBSE methodology. Only IBM Harmony-SE has such an option. IBM Harmony-SE has optional profiles for the representation and analysis of the aspects of dependability (safety, reliability and security); however, as it is mentioned in [50], the specialized tools are likely to have more capability in those domains.

The last criterion looks if the methodology is based on standard/process. OOSEM follows the ISO/IEC 15288 standard for SE lifecycle processes. This standard provides a framework of processes that should be applied to a system throughout its full lifecycle, including requirements specification, architectural design, implementation and verification [48]. IBM Harmony-SE follows the RUP process, which as well covers the full system lifecycle and provides disciplined guidelines for the roles, work products and tasks [34]. The JPL State Analysis methodology is based on the formal State Analysis process, which extends fundamental concepts from control theory and software architecture to aid in the design of complex control systems [54]. Other methodologies have not declared any dependency on the standard/process.

Table 2.1. Comparison of MBSE methodologies

	OOSEM Methodology	**IBM Harmony SE**	**SYSMOD Process**	**JPL State Analysis**	**Vitech MBSE**	**MagicGrid Approach**
Provides activities for early phase of system development	Yes	Yes	Yes	No	Yes	Yes

	OOSEM Methodology	**IBM Harmony SE**	**SYSMOD Process**	**JPL State Analysis**	**Vitech MBSE**	**MagicGrid Approach**
Suggests additional activities (other than SE)	Evaluation and optimization of system architectures	Integration with Software Implementation Model	Variant modelling; Functional Architectures	System state prediction; Performance assessment; embedded software architecture	No	Not in a current version
Modelling language	UML and SysML	SysML	SysML	SQL; additionally SysML	System Definition Language (SDL)	SysML
Supports iterative process	Yes	Yes	Yes	Yes	No	No
Validation and verification	Yes	Yes	Not in a current version	Yes	Yes	Yes
Supports security analysis	No	Yes	No	No	No	No
Based on process/ standard	Yes, ISO/IEC 15288	Yes, Rational Unified Process (RUP)	No	Yes, State Analysis process	No	No

2.4. Security Requirements Engineering

This section presents an overview of Security Requirements Engineering. The Security Requirements Engineering domain encompasses methods, processes, techniques and norms for tackling secure systems creation activity during the early stages of the system development cycle [56]. Many approaches and methods for performing security requirements engineering have been proposed in literature. Some of the approaches provide guidance for the security-related activities (e.g., SQUARE [57] and CLASP [58]), while some of them operationalize security standards (e.g., SREP is based on ISO/IEC 17799:2005 [59], and CORAS is based on ISO 31000 [60]). A detailed comparison of security requirements engineering methods was provided by Fabian et al. [61] and Mellado et al. [56].

The security requirements engineering process includes traditional requirement engineering activities such as requirements elicitation, specification and analysis. The final purpose of security requirements engineering is to prevent harm in the real world by considering security requirements as constraints upon functional requirements [62]. The most recurring term is “security requirement”, and it is worthwhile to look at how this term is treated by different authors:

1. Dubois et al. characterize security requirement as a condition over the phenomena of the environment that system stakeholders wish to make true by designing the system in order to mitigate risks [9].
2. Fabian states that the security requirement is a detailed refinement of one or more security goals, whereas the security goal refers to a part of the CIA (confidentiality, integrity and availability) model [61].
3. Salini and Kanmani agree that security requirements can be treated as a constraint on the functions of the system, and these constraints operationalize one or more security goals [62].

Respectively, in this thesis, security requirement can be considered as a more detailed statement of security goal. Security goals are most often classified into confidentiality, integrity and availability goals [61]. The ISO/IEC 13335-1:2004 standard presents the industry-proven examples of each goal [63]:

- Confidentiality is the characteristic that information is not made available or disclosed to unauthorized users, entities or external systems.
- Integrity is the characteristic of safeguarding the completeness and accuracy of assets.
- Availability is the characteristic of being accessible and usable upon demand by an authorized entity.

In the next section, it is presented how security requirement engineering is interpreted and refined in various modelling approaches for security analysis.

One remark about security and safety requirements engineering should be noted. Even though security and safety disciplines have many similarities (e.g., both are protecting assets by creating secure/safe conditions [64]), the core differences exist too [65]:

- The origin of risk: security focuses on threats (e.g., attacker hacks the aircraft in-flight entertainment system and overrides the security software), while safety considers hazards (e.g., landing gear of the aircraft fails to extend).
- The nature of the consequences: unmanaged security risks could cause harm to the system itself or to its environment. The consequences of safety risks are related to the system environment only.

In this research, only security techniques and methodologies are further analysed, except those that cover both safety and security areas (e.g., CHASSIS).

2.5. Modelling Approaches for Security Analysis

This section presents a state-of-the-art analysis of modelling approaches and techniques for security analysis and security requirement engineering[2]. The whole

[2] An analysis of UAF, CHASSIS, SysML Sec, and UML Sec methods was presented by Mažeika et al. in [88, 94].

security requirements engineering domain is overviewed in the previous section, and there, the subset of this domain is analysed in more detail.

The baseline for selecting the most popular modelling approaches and techniques for security analysis was a conceptual framework for security requirements engineering created by Fabian et al. [61] and a comprehensive survey from Kriaa et al [66]. The following graphical modelling approaches that could be used at the early stage of system design and integrated into the MBSE process are selected for further analysis:

1. Unified Architecture Framework (UAF),
2. CHASSIS Method,
3. SysML Sec,
4. UML Sec,
5. CORAS.

The formal security methods based on mathematical techniques or semiformal approaches that are based on a different graphical form than UML/SysML (e.g., Petri nets and Bayesian belief network) are not included into the research scope because different notation may include additional complexity to the MBSE model, and formal methods are usually implemented in the later phase. Moreover, the techniques used in other methods (e.g., Misuse cases in CHASSIS) are not separately analysed in this section.

A review of each security approach presents the principles and capabilities, identifies the key security concepts and main techniques and looks at how the approach can be integrated into the MBSE process.

2.5.1. Unified Architecture Framework (UAF)

UAF is an enterprise architecture framework (EAF) created by the Object Management Group (OMG) [67]. The UAF framework unifies existing military architecture frameworks (such as MoDAF, DoDAF and NAF), and, unlike the latter, it is applicable to industrial and commercial applications as well [68, 69]. Besides the demilitarization and unification of military frameworks, UAF has an additional security domain [70]. The security domain enables users to identify the security constraints and capture information assurance properties that exist during communication between resources and operational performers [67]. These information-assurance properties are aligned to NIST/DOD standards that are the base for the unified information security framework for the entire US federal government [71, 72].

UAF could be used throughout the entire system life cycle, starting with the initial concept, requirements, design specification phases, continuing with the implementation, deployment phases and finishing with operations, maintenance and disposal phases. The UAF architecture models allow users to model the complex relationships that exist between organizations, systems and systems-of-systems, and they as well enable the analysis of these systems to ensure that they meet the stakeholder’s needs. The framework enables the modelling of security as well as including cybersecurity controls [67, 68].

The UAF syntax is based upon a combination and extension of UML and SysML elements and diagrams. For example, the security processes view represents the security controls that are necessary to protect organizations, systems and information during processing. The recommended implementation for these security controls is the enhanced SysML Activity diagram [67].

The key security concepts used in UAF are security constraint, security property, security assets, security controls, risk and security impact property, probability.

The main techniques for defining security aspects are security constraints definition, risk definition, security processes definition, security structure/assets definition.

The integration to MBSE process: UAF supports the capability to model enterprise architecture (strategy, operational, personnel and resources, project and security) and, optionally, trace it with the systems-level model(s), which is modelled with SysML or UML languages.

2.5.2. CHASSIS Method

CHASSIS is a mnemonic acronym for the combined harm assessment of safety and security for information systems. The CHASSIS method allows identifying both security and safety aspects and is based on UML notation [73]. The main CHASSIS techniques are UML-based diagrams as well as traditional text-based techniques such as Hazard and Operability study (HAZOP) or security requirements specification [10, 74]. Figure 2.6. presents an overview of CHASSIS method.

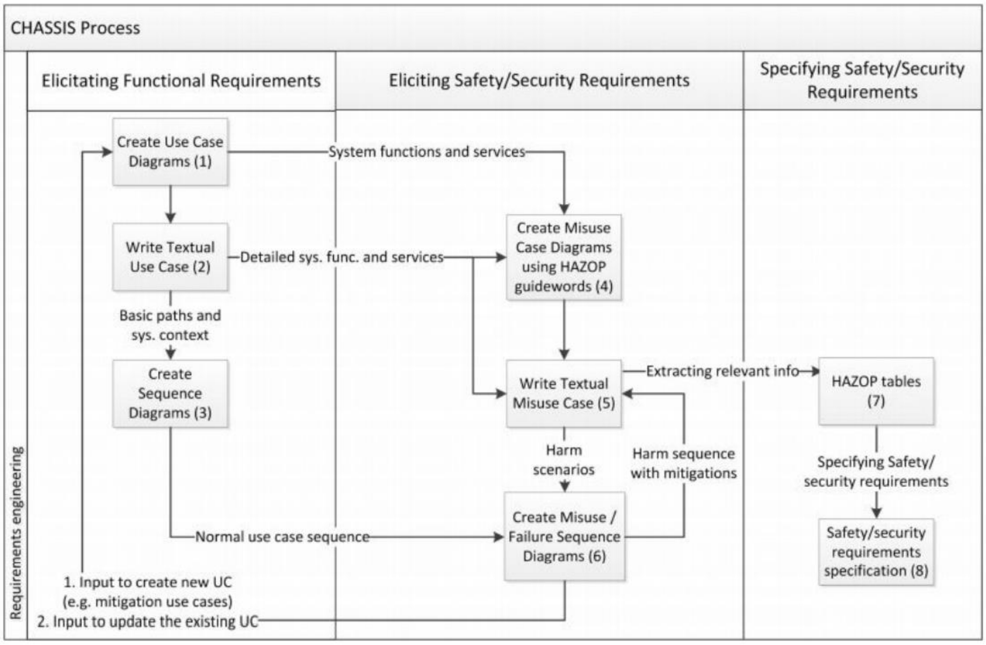

CHASSIS Process
Elicitating Functional Requirements
Eliciting Safety/Security Requirements
Specifying Safety/Security Requirements
Requirements engineering
Create Use Case Diagrams (1)
System functions and services
Write Textual Use Case (2)
Detailed sys. func. and services
Create Misuse Case Diagrams using HAZOP guidewords (4)
Basic paths and sys. context
Create Sequence Diagrams (3)
Write Textual Misuse Case (5)
Extracting relevant info
HAZOP tables (7)
Harm scenarios
Harm sequence with mitigations
Specifying Safety/ security requirements
Normal use case sequence
Create Misuse / Failure Sequence Diagrams (6)
Safety/security requirements specification (8)
1. Input to create new UC (e.g. mitigation use cases)
2. Input to update the existing UC

Figure 2.6. CHASSIS process overview diagram

There are three main steps in the CHASSIS method:

1. Eliciting functional requirements,
2. Eliciting safety/security requirements,
3. Specifying safety/security requirements.

The first two steps rely on creating and analysing UML-based diagrams (use case, sequence, misuse case, misuse sequence). Misuse case technique extends the UML use case diagram with the additional elements of misuse case and misuser. These concepts allow defining attackers and their threats to the system of interest. Moreover, two supplementary relations of threatens and mitigates allow security engineers to specify which use case mitigates misuse case or which misuse case threatens the use case. The misuse sequence diagram can be used to represent possible interactions between attacker and system that are arranged in time sequence [73].

The third steps suggest conducting results in the HAZOP table and in security/safety requirements specification [73].

The key security concepts used in CHASSIS are attack, attacker, threat, security requirement, risk and weakness.

The main techniques for defining security aspects are misuse cases, misuse case sequence diagram, HAZOP, security requirements.

Integration to MBSE process: the CHASSIS method presents a process definition, not a dedicated UML/SysML profile. As the CHASSIS method suggests using UML-based diagrams, the principles of it can be adapted to the MBSE process.

2.5.3. SysML Sec

SysML Sec is a model-driven engineering environment, which presents extended SysML diagrams for security risks as well as the methodology for creating secure real-time embedded systems. This methodology brings forward semi-formal specifications of both security and safety features and properties at various development cycle phases [75].

The SysML Sec methodology consists of three main phases [75]:

1. System analysis (based on Y-chart approach for embedded systems),
2. System design (based on V-model for software development),
3. System validation (based on model transformation into formal specifications).

The analysis phase covers the definition of security requirements and attack scenarios and serves as an identification of the main functions and candidate hardware architecture. In the system design stage, security requirements are refined with security properties, and security-related functions are defined. The validation phase allows users to formally assess whether security properties are verified. If the model is too large to be verified, model-to-code transformations are used to perform security tests [75, 76].

In the SysML Sec methodology, security requirements are based on an extended SysML Requirement diagram. A new security requirement stereotype with the property of Kind (e.g., confidentiality, access control, integrity, freshness) allows users to distinguish security requirements from functional and non-functional requirements. Attack trees can be specified with a customized SysML Parametric diagram. A Formal Dolev-Yao attacker model (for describing attacks on the protocols deployed between the components of the embedded system model) can be modelled with extended SysML Block and State Machine diagrams [75].

The key security concepts used in SysML Sec are assets, security requirement, security property, security-related function and threat.

The main techniques for defining security aspects are requirement diagrams, attack scenarios, Dolev–Yao attacker model.

Integration to MBSE process: SysML Sec was created to support all methodological stages of the design and development of embedded real-time systems. As SysML Sec uses extended SysML diagrams for capturing security concerns, the principles of it can be adapted to the MBSE process.

2.5.4. UML Sec

The UML Sec approach enables a definition of security requirements for a system under analysis with a lightweight extension of UML. As UML Sec is a lightweight extension, it does not introduce any new diagrams but provides a set of stereotypes (with tag definitions) and constraints. Security-related stereotypes allow users to specify security requirements and attack/failure scenarios with standard UML diagrams (e.g., use case, activity and sequence diagrams). The custom constraints written in OCL (Object Constraint Language) help to verify the model with formal semantics [14, 77]. In addition, the UML Sec method can be integrated with the Goal-Driven Security Requirements Engineering methodology in order to have a structured framework for secure software systems development [78].

The key security concepts used in UML Sec are security requirement, security property, attacker and attack.

The main techniques for defining security aspects are security requirements, failure/attack scenarios.

Integration to MBSE process: UML Sec is a lightweight extension for UML; thus, the security-related stereotypes can be used within SysML models; however, no default traceability or mapping with SE is defined.

2.5.5. CORAS

CORAS is a method for security risk analysis, which incorporates documentation framework, various risk assessment techniques and process description [79]. It is based on the ISO 31000 standard. This method suggests using a customized language, which is inspired by UML to model assets, risks and threats. The CORAS method consists of 8 steps for conducting security analysis [60]:

1. Initial preparation for security risk analysis;
2. The meeting between analyst and customer to clarify the security goals/requirements;
3. Identification of target, scope and main assets;
4. Approval of target, scope and main assets;
5. The security risk identification;
6. The risk level estimation;

7. The decision on which security risks are acceptable and which shall be further analysed for possible treatment;
8. Treatment identification.

The key security concepts used in CORAS are asset, security goal, risk, threat, attack, probability and vulnerability.

The main techniques for defining security aspects are security requirements, asset relationship diagram, threat diagram, HAZOP.

Integration to MBSE process: CORAS uses custom security diagrams inspired by the UML language; however, no default traceability or mapping with SE is defined.

2.5.6. Security concepts alignment

This section is dedicated to aligning all the key concepts from the analysed modelling approaches for security analysis. Table 2.2. presents security concepts with definitions, synonyms and their occurrence in the analysed modelling approaches ("+" indicates that the corresponding concept is used in modelling approach and "-" means that it is not relevant).

Table 2.2. Security concepts mapped to modelling approaches

	UAF	**CHASSIS**	**SysML Sec**	**UML Sec**	**CORAS**	**Definition**	**Synonyms**
Asset	+	+	+	-	+	Elements that can be considered as a subject for security analysis [67] Something in the system and/or its environment to be protected from negative consequences [73]	Software asset, system asset, data asset
Security constraint	+	+	+	+	+	A type of rule that captures a formal statement to define security laws, regulations, guidance and policies [67]	Security requirement, security goal
Security control	+	-	+	-	+	A safeguard or countermeasure prescribed for an information system or an organization designed to protect the confidentiality, integrity and availability of the asset's information and to meet a set of defined security requirements [67]	Security activity, safeguard, countermeasure, security-related function
Security property	+	-	+	+	+	Property or constraint on a system asset that characterizes their security need [67]	Information-assurance property
Risk	+	+	+	-	+	A statement of the impact of an event on assets [67]	-
Risk impact	+	+	+	-	-	The potential impact on system due to a specific reason (availability, integrity and confi-	Harm, consequence,

	UAF	CHASSIS	SysML Sec	UML Sec	CORAS	Definition	Synonyms
						dentiality) [67]	security impact property
Probability	+	-	-	-	+	The likelihood of risk occurrence [67]	Likelihood
Vulnerability	+	+	-	+	+	An internal fault that enables an external fault to harm the system [73]	Weakness, security constraint (in UAF)
Attacker	-	+	+	+	+	Someone or something carrying out an attack for altering the system's functionality or performance or accessing confidential information [73]	Intruder
Threat	+	+	+	+	+	Potential attack that targets system assets and that may lead to harm to the assets [9] An action carried out to harm system [73]	Attack

2.5.7. Security techniques mapping

This section introduces the mapping of the main techniques from the analysed modelling approaches for security analysis. Table 2.3. presents which security-related techniques overlap between analysed modelling approaches, how these techniques can be implemented in the SysML language and what is the purpose of each technique ("+" indicates that the corresponding technique is used in the modelling approach, and "-" means that it is not relevant).

Table 2.3. Security techniques mapped to modelling approaches

	UAF	CHASSIS	SysML Sec	UML Sec	CORAS	Implementation in SysML	Purpose
Security Requirements Definition	+	+	+	+	+	SysML Requirement diagram, SysML Use Case diagram	Captures functional and non-functional security requirements
Security Processes Definition	+	-	-	-	-	SysML Activity Diagram	Identifies security controls
Asset Structure Definition	+	-	-	-	+	SysML Block Definition Diagram, SysML Internal Block Diagram	Defines assets and allocations

	UAF	CHASSIS	SysML Sec	UML Sec	CORAS	Implementation in SysML	Purpose
Security Risk Definition	+	+	-	-	+	SysML Block Definition Diagram	Identifies and summarizes risks, risk impact, probability, etc.
Misuse Cases	-	+	-	-	-	SysML Use Case Diagram	Identifies threats and attackers
Misuse Case Sequence	-	+	-	-	-	SysML Sequence Diagram	Defines the attack sequence during an intrusion
HAZOP	-	+	-	-	+	Tabular Format	Summarizes risk and security requirements-related data
Attack/Threat Scenario	-	+	+	+	+	SysML Activity Diagram, SysML Parametric Diagram	Describes attack steps/actions
Dolev–Yao Attacker Model	-	-	-	+	-	SysML Block and State Machine Diagrams	Formally defines potential actions by an attacker

2.6. Security Risk Management

Security Risk Management can be defined *"as the systematic application of management policies, procedures, and practices to the task of establishing the context, identifying, analyzing, evaluating, treating, monitoring, and communicating security risks*" [80]. The security requirements engineering and security approaches that were analysed in previous sections focus on how to design and develop a secure system; meanwhile, security risk management covers a wider spectrum of domains that includes management of information and cybersecurity risks in an organizational context.

Comprehensive ontology for the IS security risk management was defined by Dubois et al. [9]; moreover, there was an attempt to investigate how information security risk management could help in the task of engineering a secure system [81]. There are many various security risk management standards, and in this thesis, several widely used standards are presented.

NIST 800-30. The goal of this standard is to provide guidance for conducting risk assessment activities with a special focus on the U.S. federal information systems and organizations. The NIST 800-30 guidance uses the key risk factors of threats, vulnerabilities, impact and the likelihood of threat exploitation of weaknesses in information systems to help security engineers understand and assess the current information security risks to organization and information technology infrastructures [82].

ISO/IEC 27005:2018. This standard provides guidelines for information security risk management in an organization, and it is oriented to assist the implementation of information security based on the asset, threat and vulnerability risk identification method. ISO/IEC 27005 is part of a wider ISO/IEC 27000 family of information security standards [83].

ISO 31000:2018. The standard provides guidelines on managing any type of risk faced by the organizations. It is neither industry nor sector specific, and it can be used throughout the life of the organization and can be applied to any activity, including decision-making at all levels. One of the main goals of this standard is to ensure that risk management process is efficient, effective and consistent [84].

OCTAVE (Operationally Critical Threat, Asset, and Vulnerability Evaluation). It is a risk management framework for capturing and analysing threat-related information, producing a protection strategy and mitigation plans based on the organization's security risks [85].

Figure 2.7. presents the security risk management standard processes with their steps or phases to give an overview of the similarities and differences of these standards.

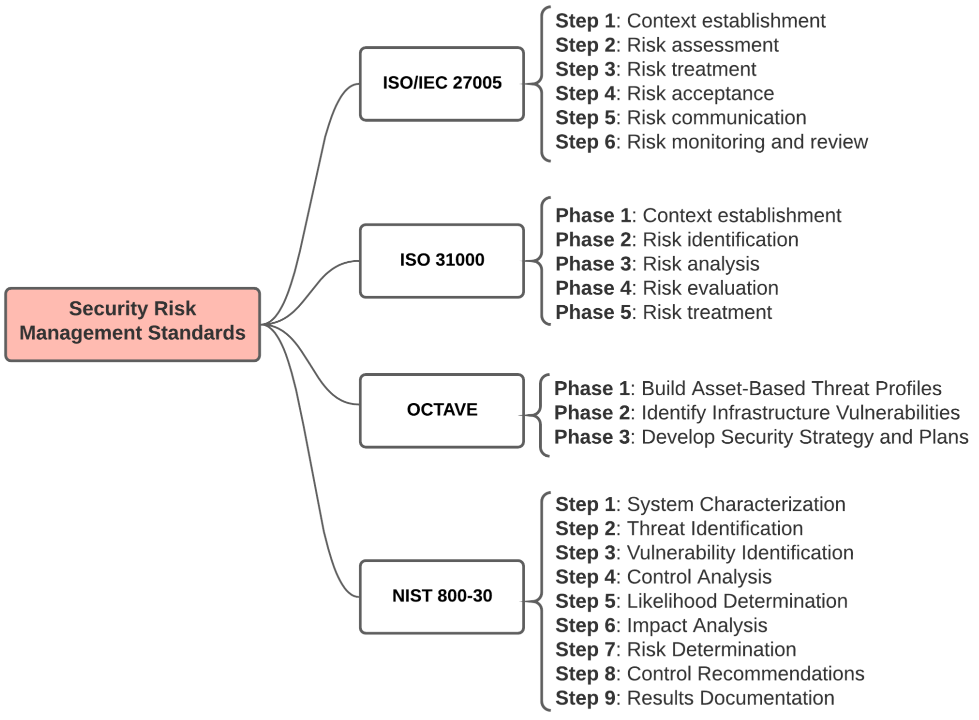

Security Risk Management Standards
ISO/IEC 27005
Step 1: Context establishment
Step 2: Risk assessment
Step 3: Risk treatment
Step 4: Risk acceptance
Step 5: Risk communication
Step 6: Risk monitoring and review
ISO 31000
Phase 1: Context establishment
Phase 2: Risk identification
Phase 3: Risk analysis
Phase 4: Risk evaluation
Phase 5: Risk treatment
OCTAVE
Phase 1: Build Asset-Based Threat Profiles
Phase 2: Identify Infrastructure Vulnerabilities
Phase 3: Develop Security Strategy and Plans
NIST 800-30
Step 1: System Characterization
Step 2: Threat Identification
Step 3: Vulnerability Identification
Step 4: Control Analysis
Step 5: Likelihood Determination
Step 6: Impact Analysis
Step 7: Risk Determination
Step 8: Control Recommendations
Step 9: Results Documentation

Figure 2.7. Security risk management standard processes with their steps/phases

2.7. Information Security Management System[3]

Information security management system (ISMS) helps security engineers to define and manage controls that an organization needs to implement to ensure that it is systematically safeguarding the confidentiality, availability and integrity of assets [86]. In this thesis, a well-established ISO/IEC 27001:2013 Security Standard is selected for ensuring that the risks would be systematically governed for the system under design.

The ISO/IEC 27001:2013 standard by the International Organization for Standardization (ISO) and the International Electrotechnical Commission (IEC) provides specific requirements for establishing, implementing, operating, monitoring, reviewing, maintaining and improving ISMS [87]. The security standard can help to develop secure complex systems in the following ways:

- It provides a step-by-step method to establish ISMS [87, 88, 81];
- It gives the best practice recommendations for information security management, risks and controls [87].

The ISO/IEC 27001:2013 4.2.1 chapter dictates the foundational steps for managing risks at a high level, and these steps can be applicable to the SE workflow at the early stage of the systems development life cycle. The activities that could be implemented in the MBSE model are as follows:

1. Define the risk assessment approach of the organization;
2. Identify the risks;
3. Analyse and evaluate the risks;
4. Identify and evaluate options for the treatment of risks;
5. Select control objectives and controls for the treatment of risks.

One of the biggest MBSE returns on investment is that by validating and verifying system characteristics early, it enables fast feedback on requirements and design decisions [15, 16]. This leads to the conclusion that the security solution should be lean as well. The ISO/IEC 27001:2013 standard recommends using the “Plan–Do–Check–Act” (PDCA) model (see Figure 2.8.), which guides that the ISMS should be continually reviewed and improved, and this principle suits the systems engineering process very well [87].

[3] Overview of ISO/IEC 27001 was presented by Mažeika et al. in [94].

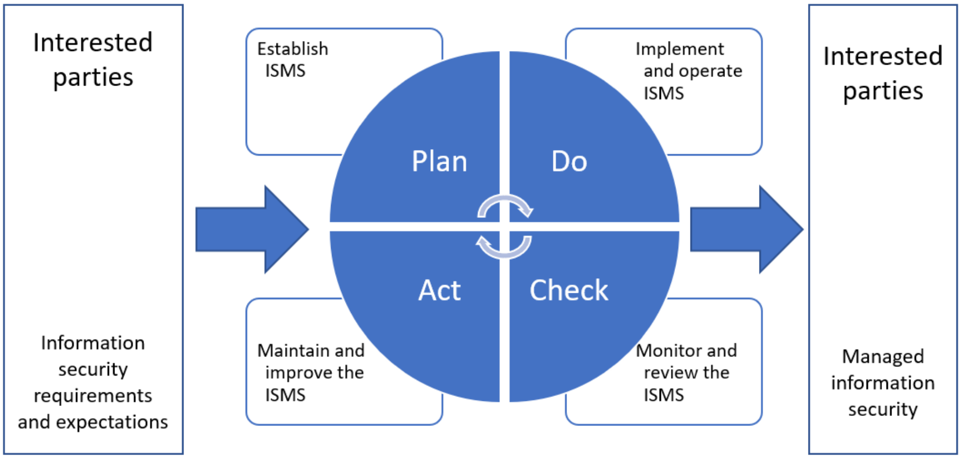

Interested parties
Information security requirements and expectations
Establish ISMS
Implement and operate ISMS
Plan
Do
Act
Check
Maintain and improve the ISMS
Monitor and review the ISMS
Interested parties
Managed information security

Figure 2.8. "Plan–Do–Check–Act" (PDCA) model applied to information security management system (ISMS) processes [87]

The ISO/IEC 27001:2013 standard gives flexibility on what techniques and methods to select for its implementation. In this thesis, the proposed MBSE method for creating secure systems follows the ISO/IEC 27001:2013 requirements for establishing the ISMS.

2.8. Summary of the Analysis

1. SE and the MBSE application encompass many disciplines that are related to complex system creation; however, the security discipline, which is crucial for a modern system, is vaguely integrated into MBSE. The opportunity to define the security aspect (with limited capabilities) was available only in one out of six leading MBSE methodologies.
2. The analysis of the leading MBSE methodologies allowed identifying common patterns that could be leveraged in developing the MBSE security method, i.e., what SE activities are performed at the early phase of the system development cycle (e.g., identifying stakeholder needs and system requirements, defining system context, modelling logical design and system behaviour); does methodology support iterative/agile modelling cycles and validation/verification capability; what standards, languages and processes are used; what additional activities are supported.
3. The literature analysis of security requirements engineering allowed to overview the whole domain and define how the security requirement should be understood in the scope of this thesis (i.e., as a refinement of security goal, whereas the security goal refers to a part of the CIA model).
4. The literature analysis of modelling approaches for security analysis allowed identifying the key security concepts and main techniques and described how these elements can be integrated into the MBSE process. The aligned security concepts and techniques serve as a core source for

the creation of domain-specific language and method.

5. The analysis of security risk management standards presented how the security risks are managed in a wider (organizational) context and what are the common phases or steps (i.e., context establishment, risk identification, risk analysis, risk evaluation, risk treatment).
6. The ISO/IEC 27001:2013 standard was selected for ensuring that the risks would be systematically governed and mitigated with the proposed MBSE security method.
7. The analysis of related works revealed that the existing security approaches, methods and techniques could potentially be leveraged in the MBSE environment if the common approach or method would be created. Because of these reasons, the solution was taken to develop the MBSE security method that would allow complete, concise, correct and consistent model security aspects of the complex systems in the CASE tool.

3. FEASIBILITY SURVEY[4]

The feasibility survey was conducted in order to validate the business needs before creating the MBSE security method. The main goal of this feasibility study is to support or deny the initial hypothesis that the MBSE would be helpful and needed during the security analysis at the early stage.

A questionnaire was sent to 10 engineering companies from the following industries: *transportation, aerospace and defence, maritime, healthcare* and *software*. The survey questions were answered by systems engineers (total: 8), a chief systems engineer and a security engineer. Below is provided a partial list of participating companies: *Rolls-Royce, ThyssenKrupp, OntoPilot, Intellerts, ProStep, Altran, 2GetThere, Dassault Systemes, Air Direct Solutions*. Other participants asked not to disclose their organization names in the research.

The first two questions were dedicated to finding out how many organization members are involved in systems engineering and how many are in security engineering activities. The results are provided in Figure 3.1.

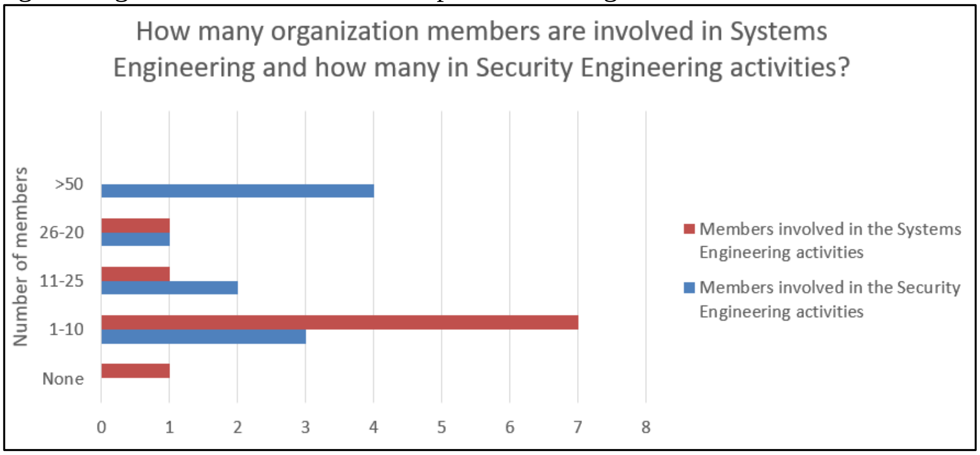


Figure 3.1. Chart presenting the number of members for systems engineering and security engineering in the surveyed organizations

As shown in Figure 3.1., the numbers of organization members that are involved in systems engineering activities are much higher than those in security engineering. Since the MBSEsec method is based on including security activities into the MBSE model, the effort of training security engineers the MBSE would be significantly lower than vice versa.

The third question was dedicated to finding out the distribution of system engineering activities. The majority of respondents perform system requirements definition and functional design activities; in addition, logical and physical design activities are widely used as well. All these activities, except physical design creation, are usually conducted at the early stage of system development. All the

[4] The feasibility survey results were published in [88].

results are provided in Figure 3.2.

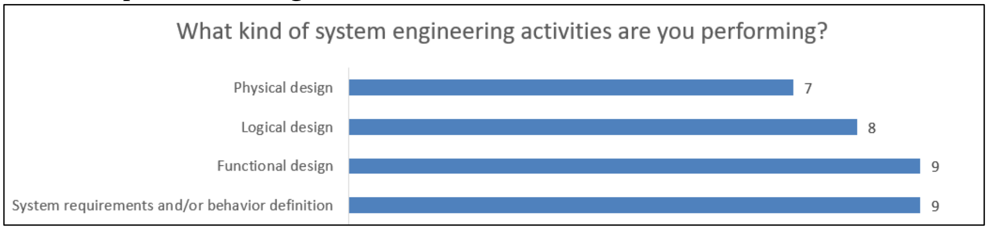


Figure 3.2. Chart presenting distribution of systems engineering activities

The fourth question was "Are the security requirements or other security artefacts represented (or linked) in your systems engineering models/documents?" In total, 6 respondents said that it was linked, 2 said that it was partially linked, and 2 respondents said that the artefacts were not linked. Moreover, the respondents were asked to elaborate more on this question; the opinions are provided below:

- No security artefacts produced. Security is approached as additional requirements for the system.
- We currently only collaborate internally in our company.
- Some system attributes that are relevant for security are modelled. Some model elements are also specifically created for security analysis purposes (networks, for example).
- Mostly by linked security requirements.
- Documentation of assets/system objects and physical and logical connection.

These answers lead to the conclusion that more than half the respondents trace security requirements with the systems engineering elements but not in a consistent way.

The fifth question was "Does your organization conform to any security standard for system design?", and 43 percent of respondents said that their process conforms to the ISO/IEC 27001 standard; all the answers are provided in Figure 3.3.

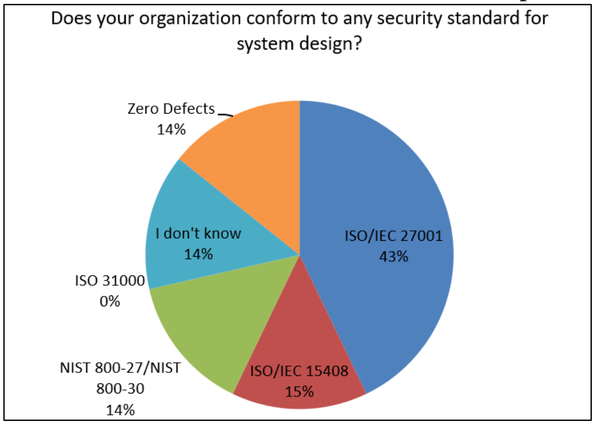


Figure 3.3. Chart of the question: "Does your organization conform to any security standard for system design?"

The next question was dedicated to finding out what techniques organizations practice for security analysis. The majority of respondents (8) rely on security requirements. The attack/threat scenarios and security processes/controls definition were practiced by 3 respondents. All the results are provided in Figure 3.4.

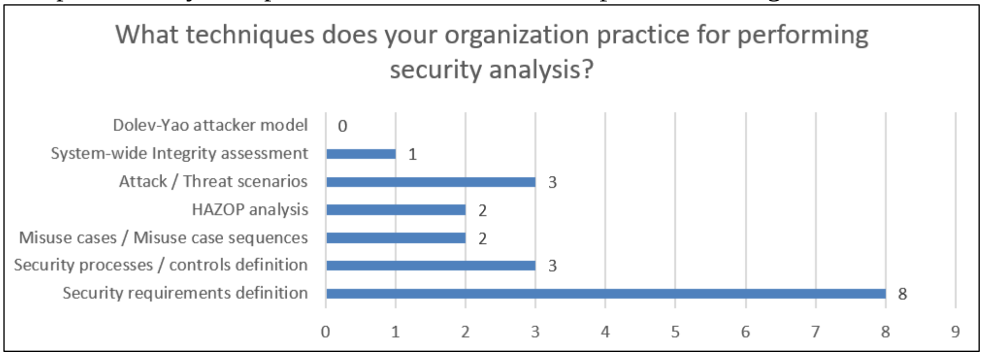


Figure 3.4. Chart representing the benefits of integrating security activities into MBSE model

The next question helped to figure out whether the security analysis integration into MBSE could bring any benefits. The majority of participants agree or strongly agree that all the listed advantages would be important. All the results are provided in Figure 3.5.

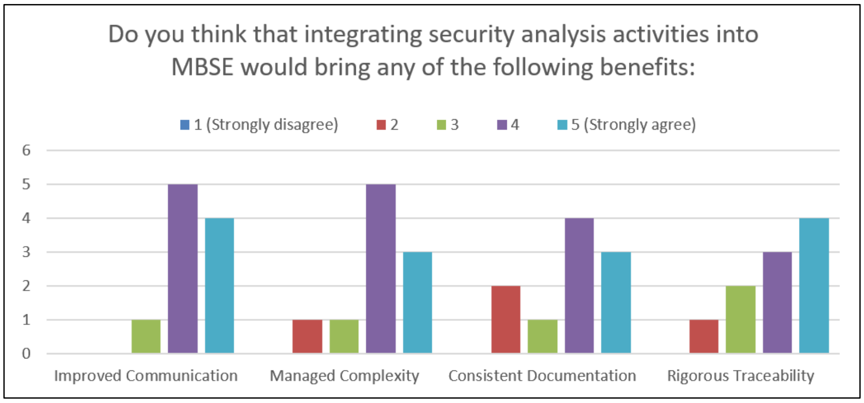


Figure 3.5. Chart of the question: “Do you think that integrating security analysis activities into MBSE would bring any of the following benefits?”

The last question was dedicated to checking which techniques would be useful for validating/verifying the security model (see Figure 3.6.).

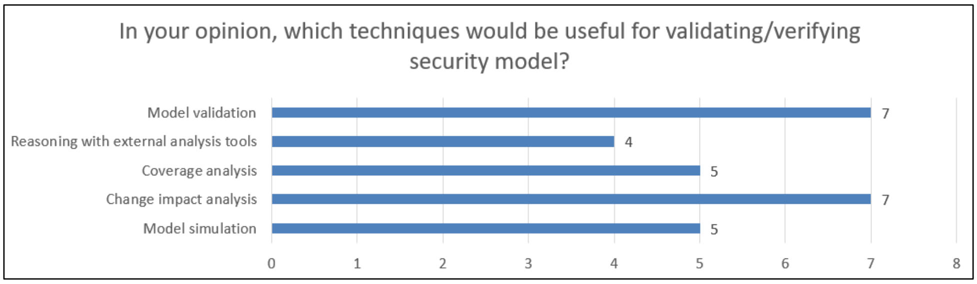


Figure 3.6. Chart representing MBSE techniques for validating/verifying security model

Seven out of ten respondents answered that the most useful techniques would be model validation (e.g., checking if the current level of risk is acceptable) and change impact analysis (e.g., analysing what assets will be impacted if the security requirement is changed). Five respondents said that the coverage analysis (e.g., checking how many risks are not linked with the security controls) and model simulation (e.g., validating if the attack scenario is executed correctly) would be useful as well.

To summarize, the feasibility survey showed that both systems engineers and security engineers acknowledge the importance and value of integrating systems and security models; however, this has not been implemented yet on a vast scale in practice or in a common way.

4. MBSE METHOD FOR CREATING SECURE SYSTEMS

This section covers the following topics:

1. The standards and tools that are used to formalize the domain-specific language and the suggested MBSE method. These include UML 2.5 profiling capability, the MOF standard, DSML definition framework and the requirements terminology by IETF RFC2119.
2. The approach on how the MBSEsec method is developed.
3. The intermediate steps towards implementing MBSEsec method (security domain model, MBSE security profile) and the MBSEsec method itself (implementation requirements, guidelines).

4.1. UML Profiles and MOF Standard for formalizing MBSE Security Method

The Meta-Object Facility (MOF) standard by OMG provides platform-independent metadata management framework and the necessary metadata services for language development and model interoperability [89]. The MOF standard is a de-facto approach for the definition of UML compatible domain-specific languages. The UML language itself is defined as a model that is based on MOF, where each UML element is an instance of one model element in MOF. Similarly, the model that is created with UML is an instance UML model [90].

The UML profiling capability enables extending metaclasses from existing metamodels in order to adapt them for domain-specific purposes. One of the most representative UML extension is the SysML language, which reuses a subset of UML 2.5 and provides additional elements to address SE requirements [41]. The principal scheme of UML and SysML interrelationship in the context of OMG meta-layer architecture is provided in Figure 4.1.

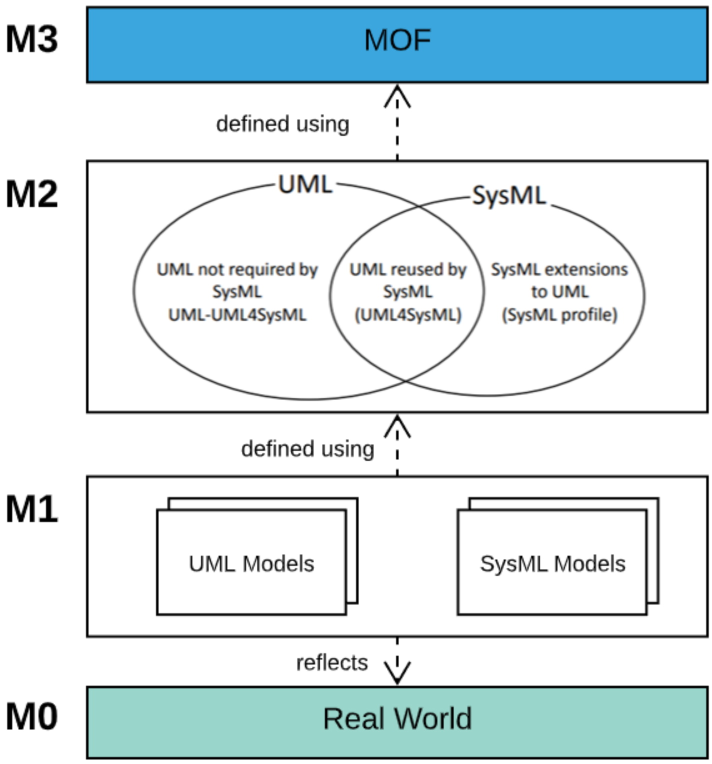

M3
MOF
defined using
M2
UML
SysML
UML not required by SysML UML-UML4SysML
UML reused by SysML (UML4SysML)
SysML extensions to UML (SysML profile)
defined using
M1
UML Models
SysML Models
reflects
M0
Real World

Figure 4.1. SysML and UML profiles interrelationship in the context of OMG meta-layer architecture [89]

In this dissertation, the MBSE security method is formalized with the UML 2.5 profiling mechanism and with the DSML definition framework, which was proposed by Šilingas et al. [91]. The UML profiling capability is not always sufficient for creating domain-specific modelling language, and the DSML framework provides an additional customization layer on top of the UML profile for virtually transforming stereotypes into new metaclasses, defining custom diagrams and verification rules. The new language defined with the DSML method out-of-the-box supports transformations, model comparison and merge, validation, code generation and other features provided by the UML CASE tool [91].

The diagram in Figure 4.2. presents the DSML modelling environment extension for security analysis in the context of OMG meta-layer architecture. The MBSE security profile is the extension of the UML profile; moreover, it inherits and extends the necessary stereotypes from the SysML profile. The diagram definitions, restrictions and rules for transforming stereotypes into metamodel are defined in the customizations package.

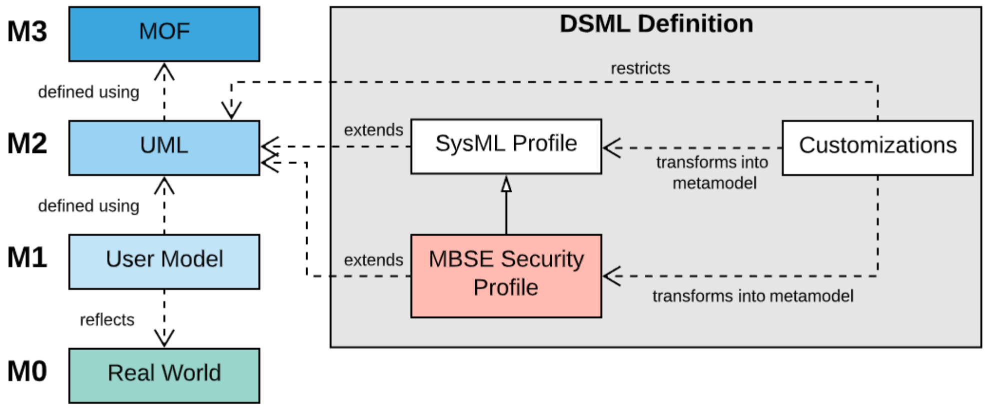


Figure 4.2. The modelling environment extension for security in the context of OMG meta-layer architecture from [91], extended by the author

4.2. Requirement terminology for the MBSEsec method implementation

Besides the UML 2.5 profiling capability and the DSML modelling environment extension, this work introduces the requirements on how the MBSE security method should be implemented. The requirements for the MBSEsec method implementation serve two purposes:

1. It makes MBSEsec method tool independent. The requirements specifically dictate what kind of diagrams, stereotypes, verification rules, etc., are needed in each security phase. Following these requirements, the MBSEsec method could be developed in any UML-based modelling tool.
2. It serves as usage guidelines. The requirements provide instructions on which security artefacts engineers should model, how the security analysis should be performed, in what sequence, what is the rationale for each phase, etc.

In this research, the requirements terminology follows IETF RFC2119 recommendations [92]. Table 4.1. presents how the significance of requirements should be interpreted in the MBSEsec method implementation.

Table 4.1. Requirement terminology by IETF RFC2119 used in the MBSEsec method implementation **[92]**

Term	Meaning in requirement text
SHALL	*This word, or the terms "REQUIRED" or "MUST", means that the definition is an absolute requirement of the specification.*
SHALL NOT	*This phrase, or the phrase "MUST NOT", means that the definition is an absolute prohibition of the specification.*
SHOULD	*This word, or the adjective "RECOMMENDED", means that there may exist valid reasons in particular circumstances to ignore a particular item, but the full implications must be understood and carefully weighed before choosing a different course.*
SHOULD NOT	*This phrase, or the phrase "NOT RECOMMENDED", means that there may exist valid reasons in particular circumstances when a particular behaviour is acceptable or even useful, but the full implications should be understood and the*

Term	Meaning in requirement text
	case carefully weighed before implementing any behaviour described with this label.
MAY	*This word, or the adjective "OPTIONAL", means that an item is truly optional.*

4.3. Approach for developing the MBSEsec method

The approach for developing the MBSEsec method is summarized in the SysML Activity diagram below (see Figure 4.3.).

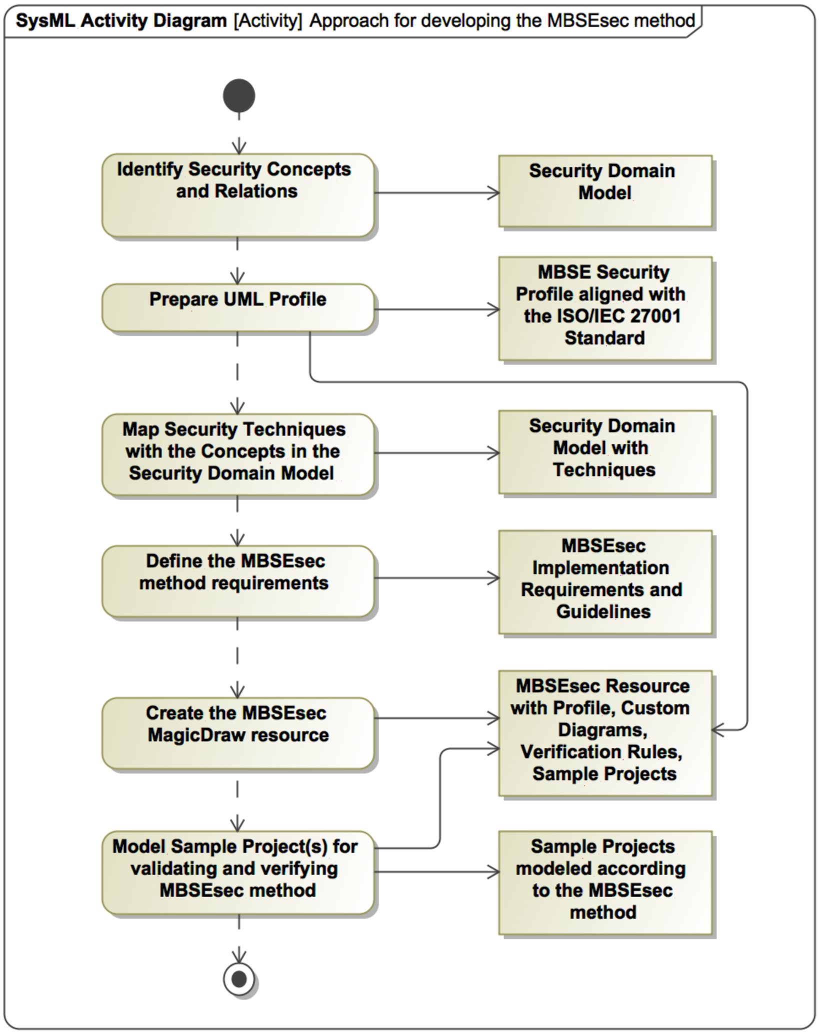

SysML Activity Diagram [Activity] Approach for developing the MBSEsec method
Identify Security Concepts and Relations
Security Domain Model
Prepare UML Profile
MBSE Security Profile aligned with the ISO/IEC 27001 Standard
Map Security Techniques with the Concepts in the Security Domain Model
Security Domain Model with Techniques
Define the MBSEsec method requirements
MBSEsec Implementation Requirements and Guidelines
Create the MBSEsec MagicDraw resource
MBSEsec Resource with Profile, Custom Diagrams, Verification Rules, Sample Projects
Model Sample Project(s) for validating and verifying MBSEsec method
Sample Projects modeled according to the MBSEsec method

Figure 4.3. The approach for developing the MBSEsec method

The first step stands for identifying security domain concepts and relations. This is a traditional modelling language design path where the key concepts of the domain are determined at first, and then, a new language or method is developed to support it [20]. The result of this activity is the Security Domain Model modelled in the UML Class diagram.

Secondly, the concepts and relations that were identified in the Security Domain Model are recreated as the stereotypes and tag definitions in the UML Profile. Moreover, the additional stereotypes are created in order the MBSE security profile would be aligned with the ISO/IEC27001:2013 security standard. The MBSE Security Profile can be considered as the *minimum viable product* to model security artefacts in the MBSE environment. The demonstration of the security profile usage of performing security analysis was presented in [88].

Next, the initial Security Domain Model is updated by introducing which security technique should tackle each concept. This mapping helps in deciding what kind of diagram, view or tool is needed to manage the security artefacts in the MBSE model.

The following step talks about defining the requirements for the MBSEsec method implementation. The requirements explicitly specify what new stereotypes, UML/SysML elements or relations shall be present in each proposed MBSEsec diagram; how each MBSEsec phase shall be modelled; what are the traceability and sequencing rules; how the model analysis shall be conducted; what is the recommended model structure. The requirement text follows IETF RFC2119 recommendations for ensuring unambiguity. Moreover, it enables implementing MBSEsec method on any UML/SysML based tool.

In the next step, the MBSEsec resource is prepared with the MagicDraw 19.0 CASE tool [93]. The resource includes the security profile, custom diagrams, verification rules and other customizations. This MBSEsec resource can be installed in any compatible MagicDraw tool.

The last step is to model sample projects for validating and verifying the MBSEsec method. These sample models are included in the MagicDraw resource as well.

4.4. Security Domain Model

The literature analysis of modelling approaches for security analysis and security concepts alignment activities that are presented in the second section serve as the core source for the domain model definition. The domain model that represents the key security concepts and their relations is modelled with the MagicDraw CASE tool in the UML class diagram (see Figure 4.4.). Initially this security domain model was introduced by Mažeika et al. in [88].

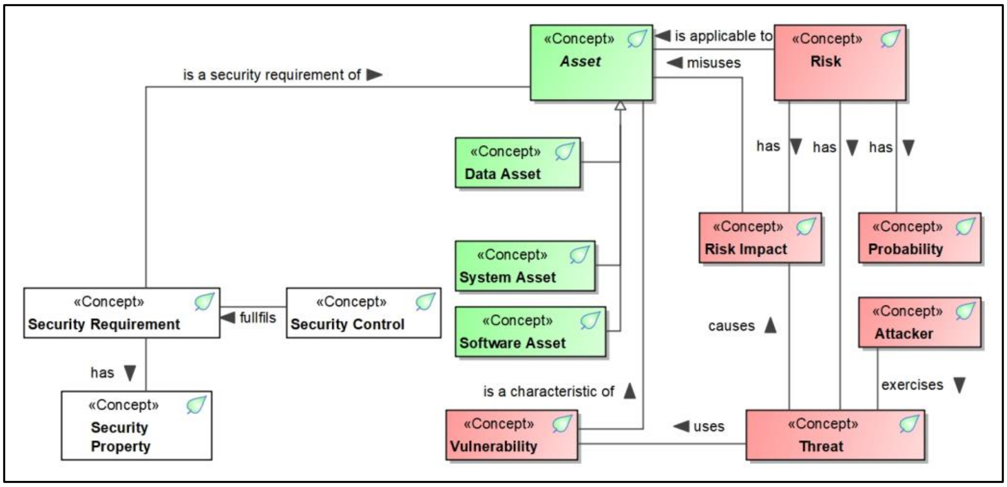


Figure 4.4. The security domain model

The security domain model has three different types of concepts:

1. Security assurance concepts (white) define concepts that allow ensuring system security or mitigating possible risks.
2. Items to be protected (green) present data and system assets that should be identified and protected.
3. Risk-related concepts (red) characterize hostile concepts and possible system weaknesses.

The security domain model represents the concepts and the relationships from the security requirement engineering field. It is the first step for defining the domain-specific language; the next section presents how the UML profile is prepared according to this domain model.

4.5. MBSE Security Profile

The security domain model serves as a key input for creating a UML profile. In addition to security domain model, the ISO/IEC 27001:2013 information security standard is used to derive additional mandatory elements and properties for establishing ISMS. The MBSE security profile diagram is presented in Figure 4.5. The profile scheme contains five groups with relevant stereotypes. Each group is aligned with the ISO/IEC 27001:2013 steps for ISMS.

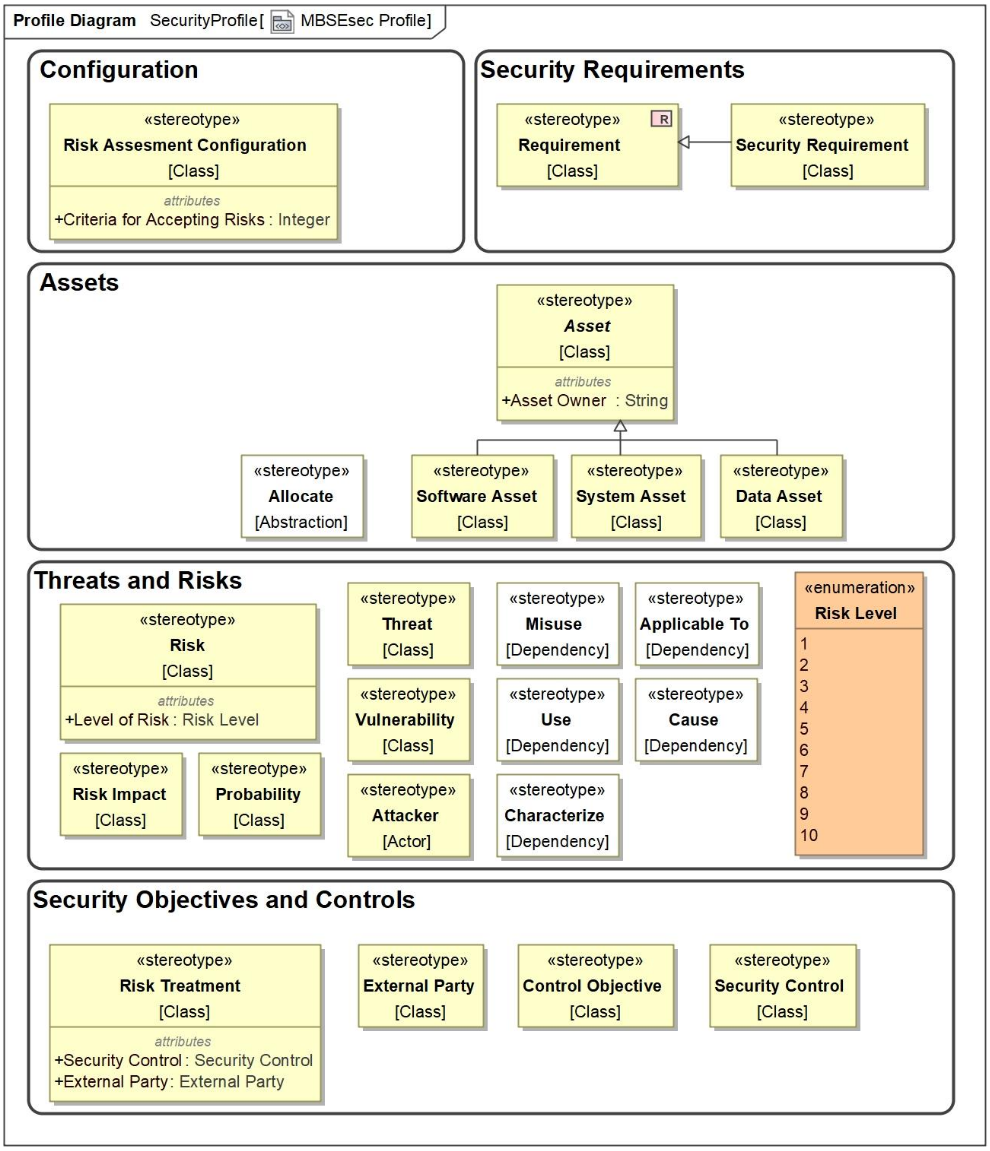


Figure 4.5. The MBSE security profile

The first profile group of Configuration contains the stereotype for "Risk Assessment Configuration" with the tag definition of "Criteria for Accepting Risks". A tag definition type is an integer number. This stereotype has derived from the ISO/IEC 27001:2013 standard, and it serves for documenting methodology of risk assessment and setting the criteria for accepting risks.

The next group is dedicated to security requirements. As a result, the "Security Requirement" stereotype, which is a subtype of the SysML Requirement element (linked with the UML Generalization relationship), is introduced.

The third group of Assets consists of stereotypes for defining and allocating system assets. According to the security domain model, one general and three specific stereotypes for Software/System/Data Assets are present. The tag definition of “Asset Owner” is inherited from the abstract Asset stereotype to three specific elements. Moreover, the dependency-based stereotype of Allocate from the SysML profile is represented in this group.

The group of “Threats and Risks” has the following risk-related stereotypes from the security domain model: Risk, Risk Impact, Probability, Threat, Vulnerability. Moreover, the dependency-based stereotypes are created for all possible relations: Misuse, Cause, Characterize, Use and Applicable To. The attribute of “Level of Risk” for Risk is added according to the ISO/IEC 27001:2013 standard. The level of risk should be set considering the risk impact and probability.

The last group of “Security Objectives and Controls” refers to the objectives and options for the risk treatment. The Security Control concept was identified during the domain analysis; moreover, ISO/IEC 27001:2013 extends this group with two related elements: Risk Treatment and Control Objective. Risk Treatment has two attributes: “Risk Control” and “Transfer to External Party”. Accordingly, the following stereotypes are created and added to this group: “Risk Treatment”, “External Party”, “Control Objective” and “Security Control”.

The next section presents all the needed elements for the MBSEsec method implementation (diagrams, customizations, OCL rules, etc.) and guidelines on how to apply it.

4.6. MBSEsec Method implementation

This section introduces the expanded security domain model and requirements for the MBSEsec method implementation. The first version of MBSEsec method was introduced by Mažeika et al. in [94].

Before defining the MBSEsec method, the initial security domain model must be updated. Firstly, the additional stereotypes that were identified in the “3.4 MBSE Security Profile” section should be represented as concepts in the security domain model (i.e., Risk Treatment, External Party, Control Objective). Secondly, MBSEsec method presents security language elements as well as activities; thus, it is worth to map the security techniques that were identified in section 2 with the security concepts from the domain model. In Figure 4.6., the security-related techniques are marked with the «Technique» stereotype (shapes filled with blue colour) and linked with the security concepts. The dependency name describes how a specific concept should be tackled with the corresponding security technique. Additionally, the techniques and concepts are grouped into two categories: *Black Box definition* and *White Box definition*. This categorization was used in MagicGrid approach [18], and it helps to decide whether the security technique/concept should be considered when the system is analysed at the abstract level with the focus on inputs and outputs (Black Box definition) or when the system parts and internal connections are analysed (White Box definition).

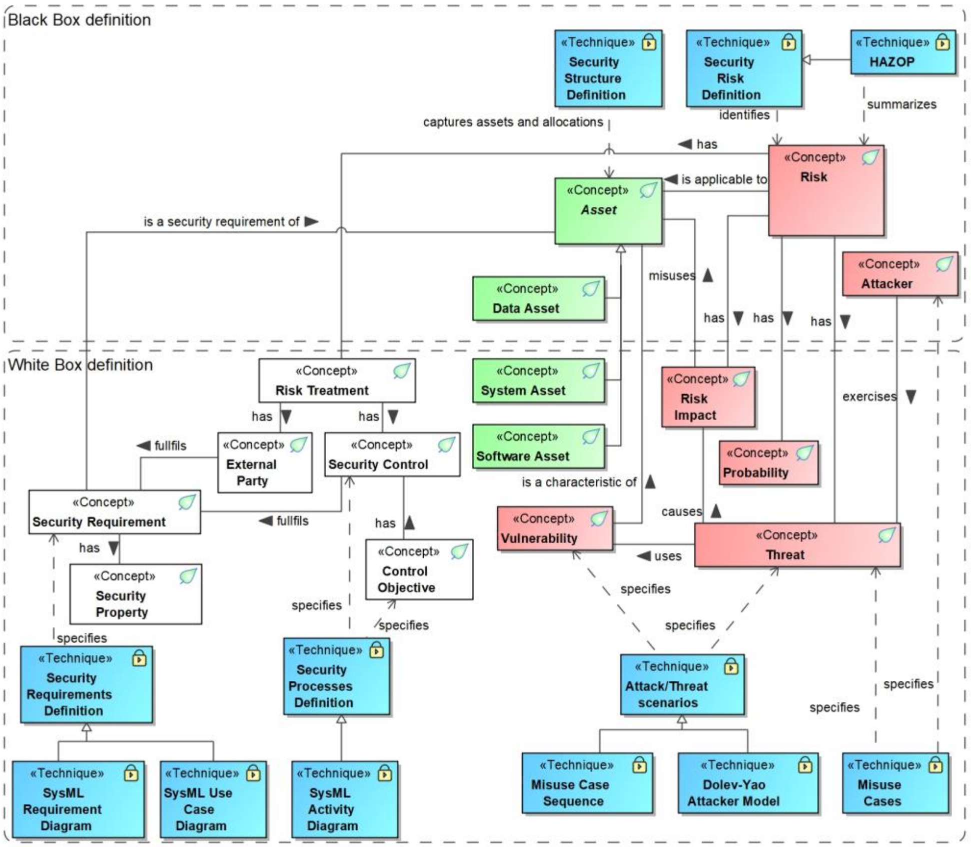

Black Box definition
«Technique» Security Structure Definition
«Technique» Security Risk Definition
«Technique» HAZOP
captures assets and allocations
identifies
summarizes
has
«Concept» Risk
«Concept» Asset
is applicable to
is a security requirement of
«Concept» Data Asset
misuses
has
has
has
«Concept» Attacker
White Box definition
«Concept» Risk Treatment
«Concept» System Asset
«Concept» Risk Impact
exercises
has
has
fullfils
«Concept» External Party
«Concept» Security Control
«Concept» Software Asset
«Concept» Probability
is a characteristic of
«Concept» Security Requirement
fullfils
has
causes
«Concept» Vulnerability
«Concept» Threat
has
«Concept» Security Property
«Concept» Control Objective
uses
specifies
specifies
specifies
specifies
specifies
«Technique» Security Requirements Definition
«Technique» Security Processes Definition
«Technique» Attack/Threat scenarios
specifies
specifies
«Technique» SysML Requirement Diagram
«Technique» SysML Use Case Diagram
«Technique» SysML Activity Diagram
«Technique» Misuse Case Sequence
«Technique» Dolev-Yao Attacker Model
«Technique» Misuse Cases

Figure 4.6. The security domain model with techniques

Some of the activities that are present in the MBSEsec method requirements directly match the name and definition of the security technique from the analysis part (e.g., SysML Requirements Diagram, Misuse Cases); some of them are derived or combined (e.g., Asset Structure Definition, Threat and Risk Definition). The third part of the techniques falls into the MBSE features category (e.g., verification rules, activity simulation, allocation matrix). The Dolev–Yao attacker model, which was identified during the literature analysis, is not included in the MBSEsec method. This technique should not be used in the early phase as it requires knowing solution details. The Dolev–Yao attacker model may be introduced in the later phases and when formal verification is needed.

4.6.1. Overview of the MBSEsec method

This section presents a quick overview of the MBSEsec method implementation (for the precise implementation requirements and guidelines see next section). Figure 4.7. presents the principal diagram in which the phases and underlying security techniques of the MBSEsec method are presented.

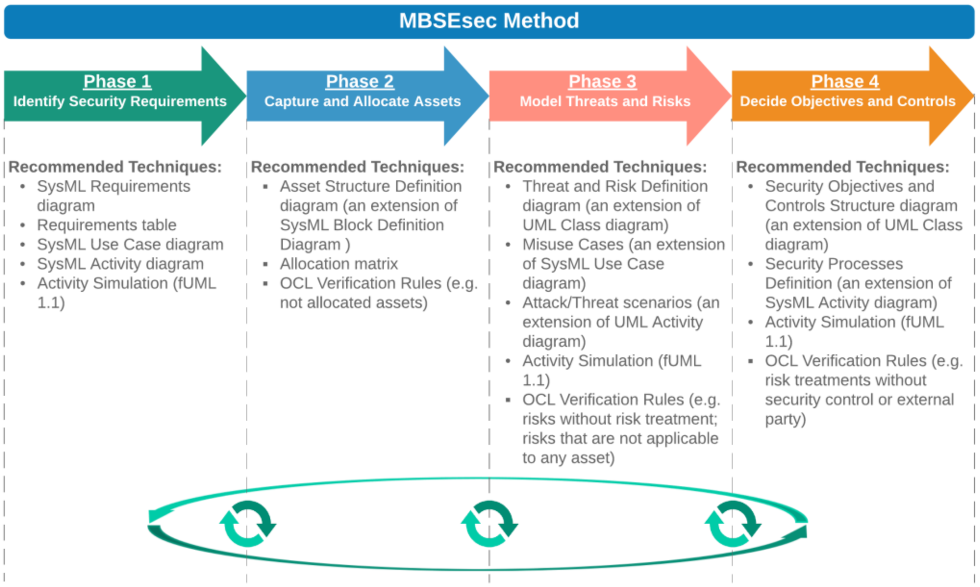

MBSEsec Method
Phase 1
Identify Security Requirements
Phase 2
Capture and Allocate Assets
Phase 3
Model Threats and Risks
Phase 4
Decide Objectives and Controls
Recommended Techniques:
• SysML Requirements diagram
• Requirements table
• SysML Use Case diagram
• SysML Activity diagram
• Activity Simulation (fUML 1.1)
Recommended Techniques:
▪ Asset Structure Definition diagram (an extension of SysML Block Definition Diagram)
▪ Allocation matrix
▪ OCL Verification Rules (e.g. not allocated assets)
Recommended Techniques:
• Threat and Risk Definition diagram (an extension of UML Class diagram)
• Misuse Cases (an extension of SysML Use Case diagram)
• Attack/Threat scenarios (an extension of UML Activity diagram)
• Activity Simulation (fUML 1.1)
▪ OCL Verification Rules (e.g. risks without risk treatment; risks that are not applicable to any asset)
Recommended Techniques:
• Security Objectives and Controls Structure diagram (an extension of UML Class diagram)
• Security Processes Definition (an extension of SysML Activity diagram)
• Activity Simulation (fUML 1.1)
▪ OCL Verification Rules (e.g. risk treatments without security control or external party)

Figure 4.7. The phases of the MBSEsec method

In Table 3.3., it is explicitly specified what new stereotypes, UML/SysML elements or relations shall be present in each proposed MBSEsec diagram.

Table 4.2. Stereotypes/elements that are available in the proposed MBSEsec diagrams

MBSEsec diagram	MBSEsec Stereotypes	UML/SysML Elements
Asset Structure Definition (an extension of SysML Block Definition diagram)	- Data Asset - System Asset - Software Asset	- Allocate
Misuse Cases (an extension of SysML Use Case diagram)	- Attacker	- Use Case (should have inverted notation, i.e., black background, white text) - Association - Include - Extend - Generalization
Attack/Threat Scenarios (an extension of UML Activity diagram)	- None	- Initial Node - Activity Final - Action - Control Flow - Decision and Merge - Fork and Join - Swimlanes
Threat and Risk Definition diagram (an extension of SysML Block Definition diagram)	- Risk - Risk Impact - Probability	- None

MBSEsec diagram	MBSEsec Stereotypes	UML/SysML Elements
	- Risk Treatment - Control Objective - Security Control - Threat - Vulnerability - Cause - Use - Misuse - Applicable to - Characterize	
Security Objectives and Controls Structure (an extension of UML Class diagram)	- Risk Treatment - Control Objective - Security Control - External Party	- None
Security Process Definition (an extension of UML Activity diagram)	- Control Objective	- Initial Node - Activity Final - Action - Control Flow - Decision and Merge - Fork and Join - Swimlanes

There are several additions: the stereotype of "Risk Assessment Configuration" can be created directly in the model or represented in the UML Class diagram. The stereotype of "Security Requirement" shall be available to apply in the SysML Requirements diagram or table.

4.6.2. Requirements for the MBSEsec method implementation

This section presents the requirements for the MBSEsec method implementation. The requirements below follow IETF RFC2119 recommendations that were described in the 3.2. section.

Prerequisites. Before starting security requirements definition and other phases for the security analysis, the information about risk assessment methodology shall be captured in the "Risk Assessment Methodology" stereotype (either as a link to the document or filling the Documentation attribute). A value for the "Criteria for Accepting Risks" attribute shall be set as an integer number. This criterion is used when executing model verification rules (e.g., checking if there are any risks that do not have risk treatment and whose risk acceptance level is higher than the acceptable level of risk).

Phase 1. Identify Security Requirements. The first phase serves for identifying the security requirements as an additional part of the functional and non-functional requirements. The "Security Requirement" stereotype, which is a subtype of the SysML Requirement, shall be used for capturing security related requirements, constraints, policies, etc. The SysML Requirements diagram or table, or both, shall be used to represent security requirements. The further security requirement refinement should be additionally done with the SysML Use Case and

Activity diagrams.

Phase 2. Capture and Allocate Assets. The second phase is dedicated to defining the objects that the organization needs to secure (assets) and allocating them to the system's parts from the logical system structure. SysML proposes two concepts for defining system structural elements, i.e., Blocks that define types and Parts that represent the usage of these blocks in a specific context. Correspondingly, the assets definition can be performed in both ways; however, the goal of this phase is to represent the structure, not their usage or internal connections. As a result, the new diagram type of Asset Structure Definition, which is an extension of the Block Definition Diagram shall be used for presenting Data, System and Software assets. The identified assets and systems blocks shall be linked with the SysML Allocation relationship (client-end is Asset, and supplier-end is Block). The allocation of these elements allows using expressions based on the Object Constraint Language (OCL) or other programming languages for running quantitative model verification, i.e., finding all system blocks that are not allocated to any asset element.

Phase 3. Model Threats and Risks. This phase represents two aspects of risk and threat definition, i.e., behavioural and structural.

The behavioural risk and threat definition shall be specified with the extended Use Case diagram for identifying Misuse Cases and the SysML Activity diagram for modelling Attack Scenarios. The Attacker stereotype, which represents the role of hostile actor, and use-case, which represents the unwanted usage of the system (misuse), shall have notation with inverted colours in the misuse technique (i.e., background filled black; the examples are presented in the Case Studies section). The attack scenario that represents a flow of actions dedicated damaging the system's integrity, availability and confidentiality shall use the standard notation of the UML/SysML Activity diagram.

The structural threat and risk definition shall be specified with the new Threat and Risk Definition diagram. This diagram shall be based on the UML Class diagram. It shall be able to create and link the following elements: Risk, Risk Treatment, Security Control, Control Objective, Risk Impact, Probability, Threat and Vulnerability (the linking rules shall be similar to the ones presented in Figure 3.7). Additionally, a HAZOP style table may be used to summarize risk-related information.

Phase 4. Decide on Objectives and Controls. The last phase is dedicated to defining security control objectives and security controls. A new "Security Objectives and Controls Structure" diagram (an extension of the UML Class diagram) shall be used for defining elements of security objectives and controls. The "Security Objectives and Control Structure" diagram shall allow to create the following elements: Control Objective, Risk Treatment, Security Control, External Party, Transfer to External Party and Apply Control. Risk Treatment may have more than one Security Control or External Party.

The standard UML Activity Diagram shall be used for identifying workflow or algorithm for security control. The security control should be modelled according to the fUML1.1. standard; this would allow security engineers to simulate and verify it.

Sequencing. Intuitively, the MBSEsec method begins with the first phase and finishes with the last; however, the phases of MBSEsec should not necessarily be conducted consecutively. The PDCA model is recommended, in which the outcomes of MBSEsec phases should be continuously reviewed and updated after each phase, i.e., it is recommended to update the Risk Treatment in the Threat and Risk Definition diagram (in Phase 3) after identifying the Security Controls in Phase 4.

Traceability. Traceability among different security phases (and within phases) is a very important aspect of the MBSEsec method. The security model shall be continuously maintained, i.e., the changes need to be managed; impact analysis needs to be performed, etc. This is not possible without having specified traceability. In the SysML language, there are four different types of relationships:

1. Direct relationship, e.g., refine, trace, allocate, derive;
2. Metaproperty, e.g., subject, owner of an element;
3. Composition (part property for block and call behaviour action for activity);
4. Derived/Implied relationships (indirectly associated elements).

The recommended relationship types and mapping rules for the MBSEsec method are depicted in Figure 4.8.

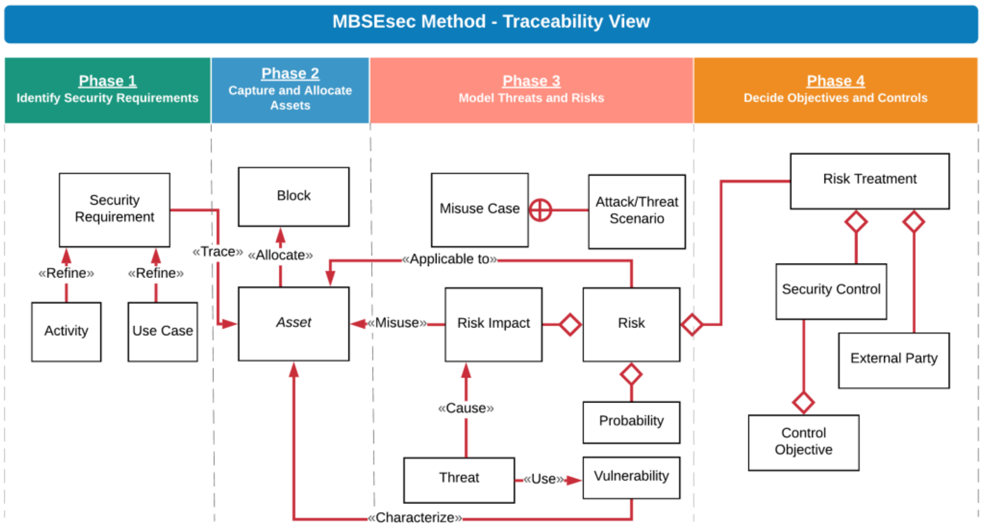


Figure 4.8. Traceability in the MBSEsec method

Model analysis. The automated quantitative model analysis, powered by the MBSE tool, enables modellers to get answers about the security model completeness, correctness and other questions.

Table 4.3. presents recommended rules for model analysis and supporting algorithms (the provided rules are not limited to this list, and additional verification rules can be introduced by any user). These algorithms can be formatted to specific programming language syntax (e.g., OCL 2.0, JavaScript) and used as metrics or verification rules to evaluate the current state of the model.

Table 4.3. Questions and algorithms for quantitative model analysis

Rule ID	Question/Verification Rule	Algorithm
R1	Are there any risks that do not have risk treatment and whose risk acceptance level is higher than an acceptable level of risk defined in Risk Assessment Configuration?	`SELECT all instances of Securi-tyProfile::Risk WHERE (SecurityProfile::Risk does not have property whose type is instance of SecurityPro-file::RiskTreatment AND SecurityPro-file::Risk::RiskLevel is greater than SecurityPro-file::RiskAssessmentConfiguratio n::CriteriaForAcceptingRisks)`
R2	Are there any Risk Treatments that do not have assigned Security Control or External Party?	`SELECT all instances of Securi-tyProfile::RiskTreatment WHERE (SecurityPro-file::RiskTreatment does not have property whose type is in-stance of SecurityPro-file::ExternalParty OR Securi-tyProfile::SecurityControl`
R3	Are there any risks that are not applicable to any asset?	`SELECT all instances of Securi-tyProfile::Risk WHERE NOT EXISTS (dependency of SecurityProfile::ApplicableTo between SecurityProfile::Risk AND SecurityProfile::Asset)`
R4	Are there any system blocks that are not al-located to assets?	`SELECT all instances of SysML::Blocks WHERE NOT EXISTS (dependency of SysML::Allocate between SysML::Block AND SecurityPro-file::Asset)`

Model structure. A well-organized model is easier to read, understand and maintain. The SysML language has the Package element for structuring diagrams and elements (including other packages). The MBSEsec method recommends using the hierarchical structure divided by different phases as it is shown in Figure 4.9. Two additional packages are dedicated to capture “Risk Assessment Configuration” and contain elements for “Model Analysis” (e.g., change impact maps, metric tables).

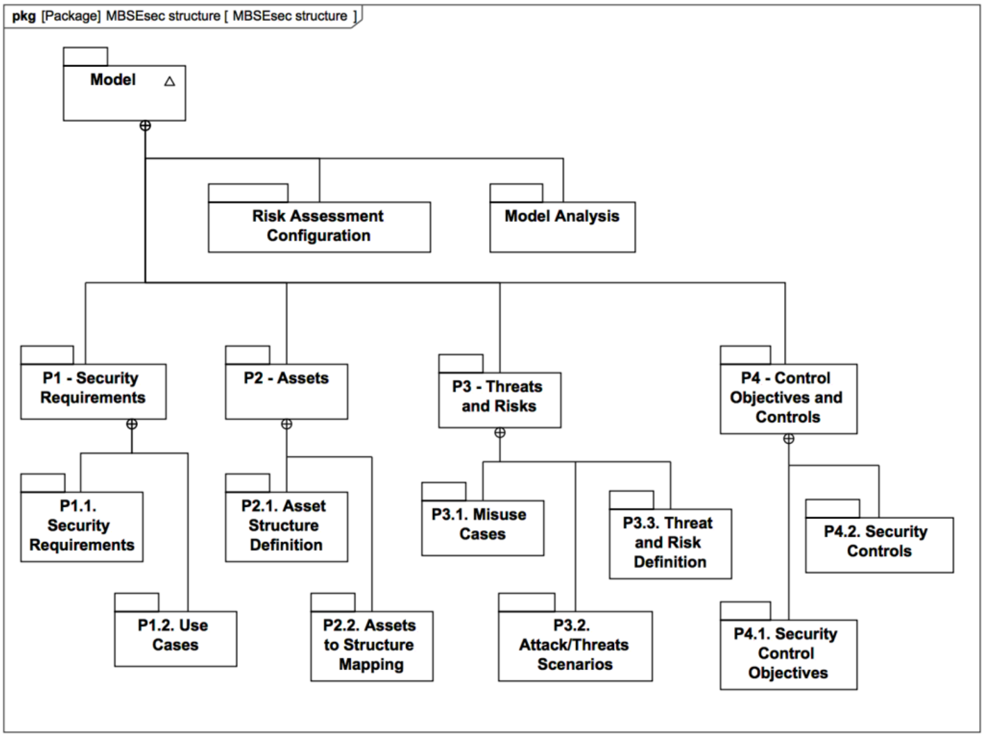

pkg [Package] MBSEsec structure [MBSEsec structure]
Model
Risk Assessment Configuration
Model Analysis
P1 - Security Requirements
P2 - Assets
P3 - Threats and Risks
P4 - Control Objectives and Controls
P1.1. Security Requirements
P1.2. Use Cases
P2.1. Asset Structure Definition
P2.2. Assets to Structure Mapping
P3.1. Misuse Cases
P3.2. Attack/Threats Scenarios
P3.3. Threat and Risk Definition
P4.1. Security Control Objectives
P4.2. Security Controls

Figure 4.9. Recommended model structure by the MBSEsec method

4.7. Evaluation

This section presents the evaluation of the proposed MBSE method for creating secure systems. The validation and evaluation consisted of several phases. Firstly, the expert evaluation was conducted in order to check if the suggested MBSEsec principles, concept and approach of integrating security analysis into MBSE are viable. Then, two real-world systems were modelled using the proposed MBSE security method in which the viability for cyber-physical and software systems were presented. Finally, these case studies were experimentally tested against four criteria: completeness, correctness, conciseness and consistency.

4.8. Qualitative evaluation of the MBSEsec method[5]

The qualitative evaluation of the MBSEsec method was done by surveying experts from the MBSE, engineering and academic fields. The goal of the survey was to validate the viability of the principles, concept and approach of integrating security analysis into MBSE.

The surveyed experts were from various engineering organizations (transportation, aerospace and defence, maritime, healthcare and software), in total 16, and from academia, in total 4. A partial list of organizations includes: *SPEC*

[5] The survey results were presented by Mažeika et al. in [94].

Innovations, *Oticon Medical*, *OAG Aviation Worldwide*, *Ocado*, *Dassault Systemes*, *Booz Allen Hamilton*, *University of Arizona*, *University of Detroit Mercy, Kaunas University of Technology*. Other participants asked not to disclose their organization names in the research.

The respondents assigned themselves to the following disciplines:

- Systems Engineering: in total 10,
- Software Engineering: in total 6,
- Requirements Engineering: in total 3,
- Mechanical Engineering: in total 1.

All the participants were practicing MBSE, and 60% of them were doing this for more than 5 years.

In order to confirm the needed number of experts of the survey, the methodical assumptions from the classical theory of tests were leveraged. According to [95], the confidence of the aggregated individual judgments and the number of experts have a steep decreasing non-linear dependency. In other words, the accuracy of the evaluations of a small expert group of aggregated expert assessments in models with equal weights is almost equal to the accuracy of large expert group evaluations. Libby et al. [95] demonstrated that the accuracy of the survey is higher than 90% when there are at least 7 experts, the precision after inviting more experts increases only fractionally. For the MBSEsec method evaluation, there were surveyed 20 experts, which in current circumstance is a sufficient number of respondents.

The survey started by looking at the participants' current work principles related to the system development. The respondents were asked to answer if they follow agile modelling practices with fast learning and validation cycles or they prefer a linear approach (e.g., waterfall methodology). Most (75%) of the respondents use agile methods, 15% use a mixed approach and 10% prefer the waterfall methodology. As the MBSEsec method is based on the PDCA model, which propagates fast creation and validation cycles, it should match the majority of respondents' practices.

The next question checked if the experts capture requirements in textual form or they additionally use UML/SysML diagrams (e.g., Use Case, Activity diagram). In fact, 53% of respondents use a textual form plus Use Case/Activity diagrams. Only 14% of respondents specify requirements in textual form and/or SysML requirements diagram. The distribution of expert opinions is provided in Figure 4.10. Moreover, the respondents were asked to elaborate more on this question, there are their opinions:

- Strong preference for properties/state machines/etc. to define requirements and use text as the last resort.
- Domain-specific modelling language (with requirements, security requirements, etc.).
- Requirements are generated from SysML architecture models.
- We do a textual form and LML (Lifecycle Modelling Language).

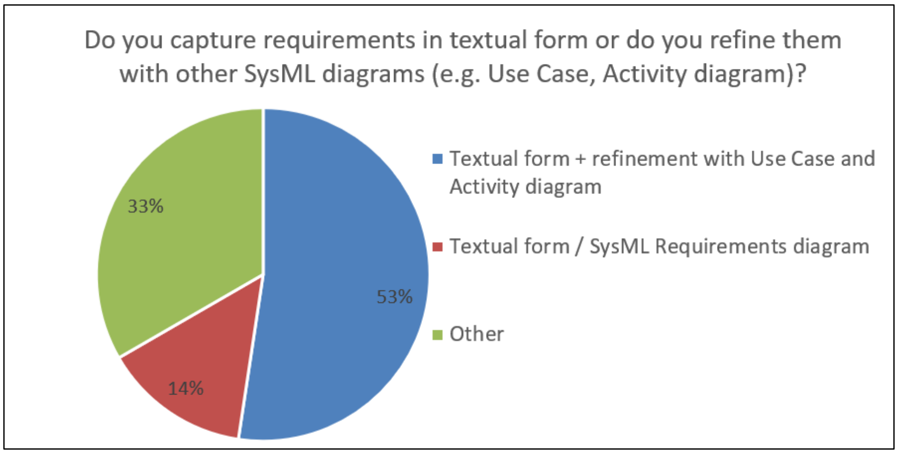


Figure 4.10. Chart representing the importance of each MBSEsec method phase

The last question related to the work principles was "Do you identify security-related requirements together with other functional and non-functional requirements?" More than the half of respondents (60%) captures security requirements, and this leads to the conclusion that the considerations of system security are quite commonly made at the early stage of system development.

Next, the respondents were asked to evaluate the importance of security mitigation phases from the MBSEsec method. Most of the participants said that the identification of parts of the system that could be vulnerable is very important or important. More than half of the participants agreed that all the other security phases are important or very important as well. All the results are provided in Figure 4.11.

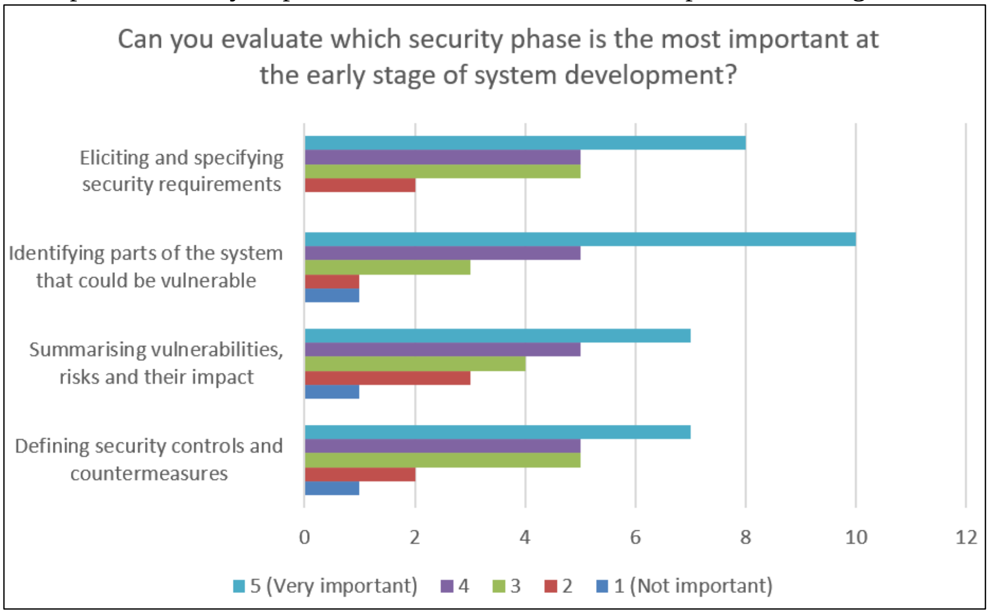


Figure 4.11. Chart representing the importance of each MBSEsec method phase

Furthermore, the experts were asked to share any other security phases/activities (not mentioned in the previous question) that should be conducted at the early stage of the system's development; their opinions are as follows:

- It is not only important to identify parts themselves that could be vulnerable but also their interaction/communication/links with other parts.
- Information exchange analysis.
- Embed security controls into the processes at all levels.
- Calculate vulnerability scores (e.g., CVSS), link security aspects with the rest of the design.
- Threat model.

In terms of MBSE tools that would be the most suitable for running combined Systems and Security Engineering analysis, participants tended to agree that Representing information in different views (diagrams, tables, matrices) and the Single source of truth are the most important. According to the experts, the least important tool is Automated document generation. The detailed answers are provided in Figure 4.12.

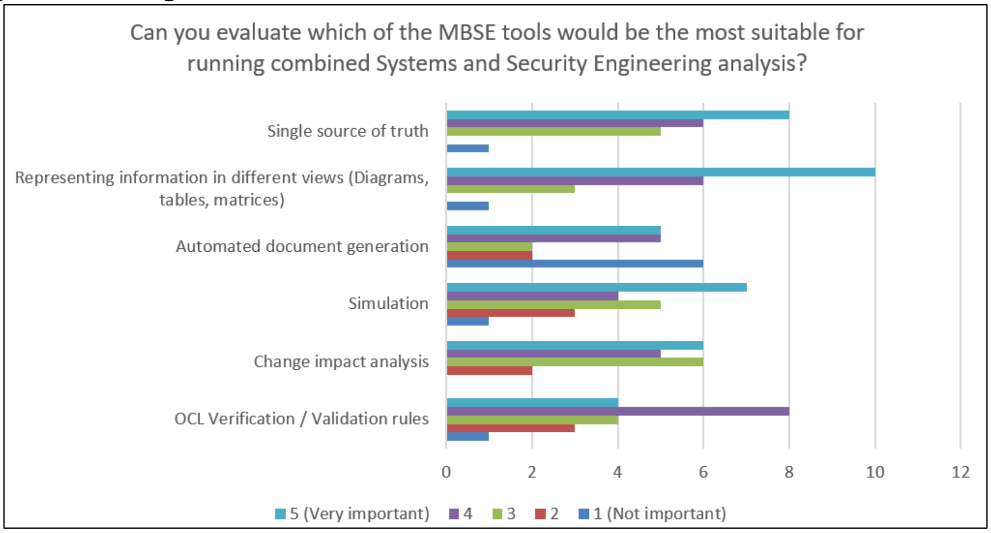


Figure 4.12. Chart representing the importance of MBSE tools for running combined Systems and Security Engineering analysis

The experts were asked to compare their efficiency when they moved from document-based system engineering to model-based system engineering. Most of the respondents (65%) said that their productivity increased, and the remaining said that productivity did not change, or it decreased. All the results are provided in Figure 4.13.

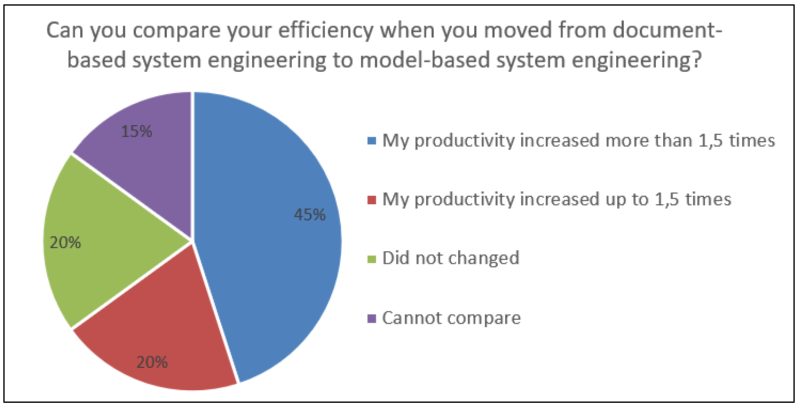


Figure 4.13. Chart of the question: “Can you compare your efficiency when you moved from document-based system engineering to model-based system engineering?”

The next question was “Did your work quality improve when you moved from document-based system engineering to model-based system engineering?” The majority of participants agreed that all the factors (Completeness, Consistency, Communication, Less defects) were improved. All the results are provided in Figure 4.14.

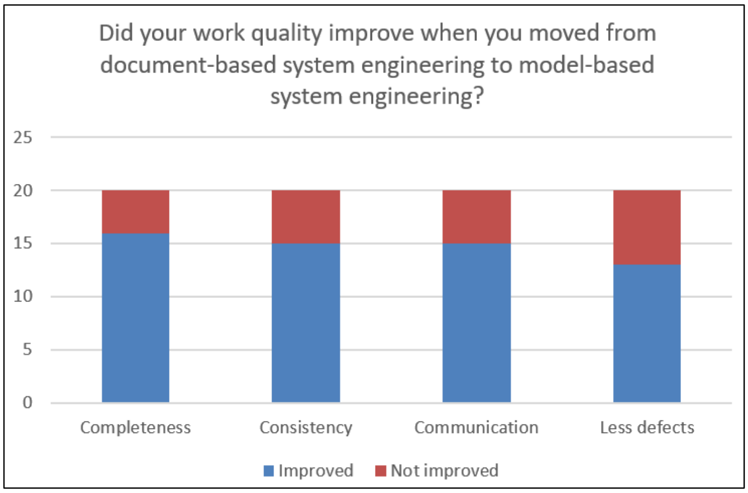


Figure 4.14. Chart of the question: “Did your work quality improve when you moved from document-based system engineering to model-based system engineering?”

The last evaluation objective was to find out the learning time required to start using the MBSEsec method. For this, the experts were asked to approximately estimate how long it would take to learn to model 5 new UML/SysML-based diagrams that have 14 custom elements and 6 relations with the assumption that the experts know the domain knowledge very well (i.e., do such analysis in Excel in their daily work). 61% of respondents answered that it should take 2 to 5 days, while 33% indicated that it would take less than 2 days and 5.5% said that it would take up to 2 weeks. Moreover, the participants commented more on this topic:

- It depends on how closely those new elements of the language map to the customer domain.
- Assuming mastery in SysML, it will take 2 months to learn and use the new elements in the production.
- Based on our experiences, it is best to have directly security (and safety) related language concepts rather than general SysML or UML concepts. The closer the mapping to the problem, the easier it becomes to use and introduce.

Finally, a rank-based nonparametric test of Kruskal-Wallis H was used to determine if there are statistically significant differences between experts with different experiences (independent variable) and required time to learn to model new UML/SysML based diagrams (dependent variable). The Kruskal-Wallis H test allows analysing two or more groups of an independent variable on a continuous or ordinal dependent variable [96]. The Kruskal-Wallis H test was conducted with the Stata statistical software package [97]. Two experts' opinions were not taken into analysis because they were not able to provide a specific time range for this question.

Table 4.4. presents the summarized results of Kruskal-Wallis equality-of-populations rank test.

Table 4.4. Table representing the results of Kruskal-Wallis equality-of-populations rank test

Experience	**Observations**	**Rank sum**
Less than a year	2	24.00
1 to 3 years	4	39.50
3 to 5 years	2	15.50
More than 5 years	10	92.00
$\chi 2$= **0.705** with 3 d.f.		
Probability (p) = **0.8721**		

A Kruskal-Wallis test revealed that there was a statistically insignificant difference in learning time between the four groups with a probability of 87.21%. The null hypothesis that population medians are all equal cannot be rejected.

To summarize, the qualitative evaluation showed that the experts are extensively using various range of UML/SysML diagrams for specifying system requirements, and more than half of them are capturing security requirements. The participants see an importance in mitigating security risks with MBSE tools at the early stage of the system development life cycle. All the phases from the MBSEsec method (with some additions) are relevant and important for the respondents. The required learning time to learn additional security concepts is relatively low for practicing MBSE users, and it as well brings higher efficiency and better work quality.

4.9. Experimental evaluation of the MBSEsec method

The experimental evaluation of the MBSEsec method was conducted on two different real-world projects. The first one represents the Hybrid Sport Utility Vehicle, which stands for the cyber-physical system, and the second represents how the MBSEsec method could help in rebuilding the legacy Flight Status software system.

The experiment conditions were as follows:

- Two real-world models were created using the MBSEsec method with the MagicDraw 19.0 CASE tool and SysML and Simulation toolkit plugins.
- The following criteria were checked for these models:
    1. Method completeness, evaluated by checking if all the artefacts that are mandatory for defining security-related documentation (i.e., comparing with the requirements for establishing ISMS according to the ISO/IEC 27001:2013 standard) were created in the MBSE model.
    2. Method conciseness, evaluated by checking how many of the MBSEsec diagrams, elements, relationships were not necessary for defining the security aspect.
    3. Correctness, checked if the security aspect can be correctly modelled with the MBSEsec method, evaluated by running UML correctness constraints validation suite, which is available with the MagicDraw 19.0 CASE tool [98].
    4. Consistency, checked if the model consistency is ensured between security phases and between the system model and security aspect.

4.9.1. Hybrid Sport Utility Vehicle-Power Control Unit

The first case study presents how the MBSEsec method can be used in defining and analysing security aspects while designing automobile. For this, the Hybrid Sport Utility Vehicle (HSUV) MBSE model from the OMG SysML specification [41] was selected as a baseline and expanded with all the MBSEsec phases. The first version of HSUV-Power Control Unit case study was presented by Mažeika et al. in [94].

Problem summary: a modern vehicle is a subject to cyber-attacks through its various network interfaces to the public network infrastructure as well as its direct exposure to the open physical environment [99]. As identified by [100], there are many vehicle parts and components that can be attacked (see Figure 4.15.); nevertheless, this case study focuses on the Power Control Electronic Control Unit (ECU), and it introduces how the security issues for this ECU can be identified, analysed and mitigated.

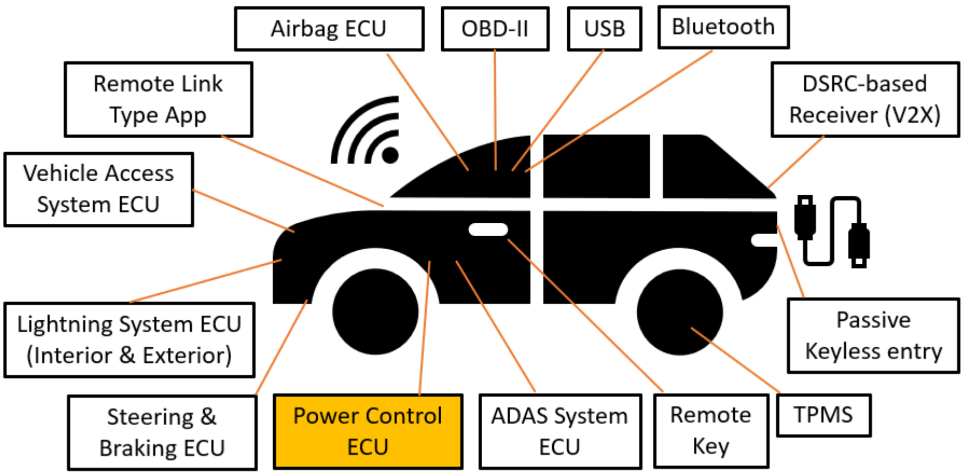
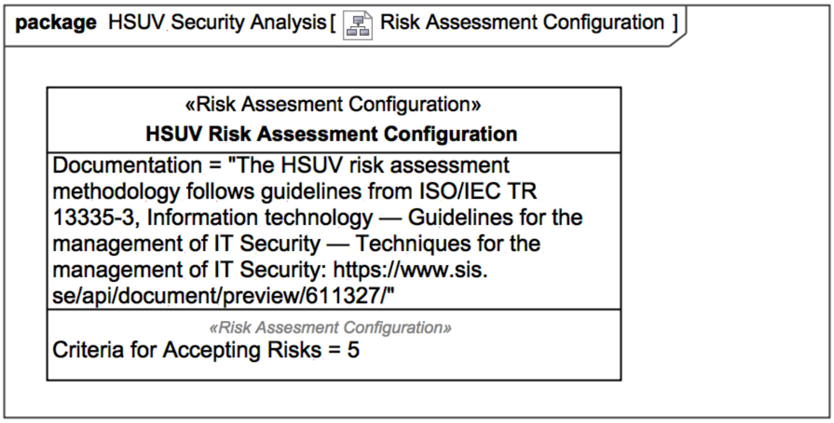


Figure 4.15. Potential attack surfaces in the HSUV

Before starting the security analysis, security engineers must ensure that the risk assessment methodology and criteria for accepting risks are decided and documented. The methodology should be captured in the "Risk Assessment Configuration" model element as documentation or link to the document. The "Criteria for Accepting Risks" should be set as an integer number. For this case study, the "Criteria for Accepting Risks" is set to 5, and the documentation part indicates that the guidelines from ISO/IEC TR 13335-3 should be followed (see Figure 4.16.) [101].



package HSUV Security Analysis [Risk Assessment Configuration]

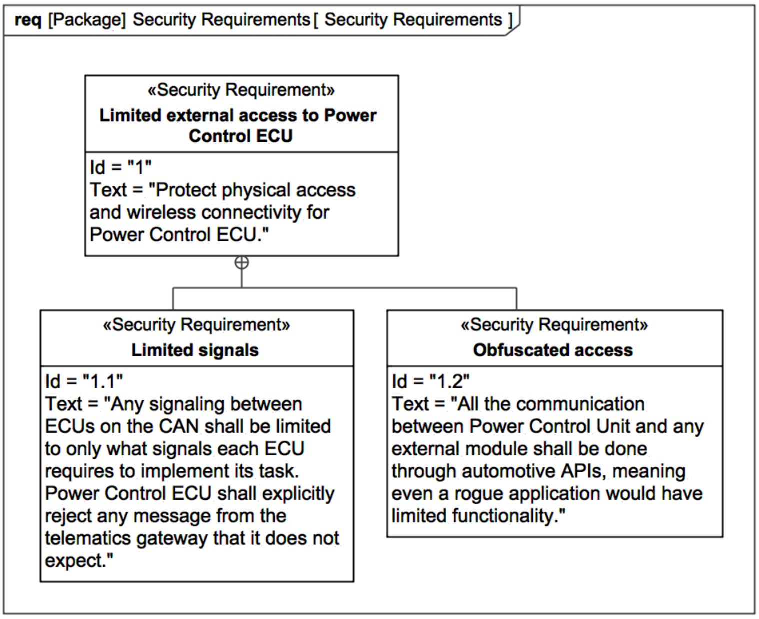
«Risk Assesment Configuration»
HSUV Risk Assessment Configuration

Documentation = "The HSUV risk assessment methodology follows guidelines from ISO/IEC TR 13335-3, Information technology — Guidelines for the management of IT Security — Techniques for the management of IT Security: https://www.sis.se/api/document/preview/611327/"

«Risk Assesment Configuration»
Criteria for Accepting Risks = 5

Figure 4.16. HSUV Risk Assessment configuration

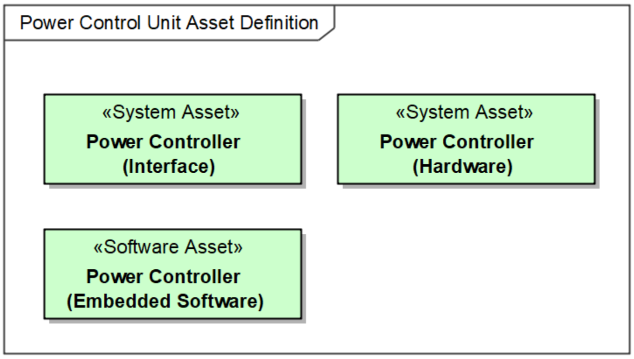
The first phase of "Identify Security Requirements" says that the security requirements shall be identified, captured and refined in the MBSE model. Ideally, engineers who are working in the security requirements engineering discipline should combine expertise in security, domain and requirements engineering fields to provide a foundation for developing a secure system [102]. Depending on the novelty of the system, the expertise of the security engineer and the security requirements engineering methodology, the security requirements can be very precise or more abstract. In this case study, for the Power Control ECU part, the explicit security requirements dictate that external access to the ECU shall be limited in two ways, i.e., by obfuscating access and rejecting unrecognized messages (see the security requirements diagram in Figure 4.17.). Optionally, the security requirements can be refined with the SysML Use Case and Activity diagrams (in this case study, not refined). Moreover, the security requirements shall be linked with the assets via the Trace relationship.



req [Package] Security Requirements [Security Requirements]
«Security Requirement»
Limited external access to Power Control ECU
Id = "1"
Text = "Protect physical access and wireless connectivity for Power Control ECU."
«Security Requirement»
Limited signals
Id = "1.1"
Text = "Any signaling between ECUs on the CAN shall be limited to only what signals each ECU requires to implement its task. Power Control ECU shall explicitly reject any message from the telematics gateway that it does not expect."
«Security Requirement»
Obfuscated access
Id = "1.2"
Text = "All the communication between Power Control Unit and any external module shall be done through automotive APIs, meaning even a rogue application would have limited functionality."

Figure 4.17. Security requirements for Power Control ECU

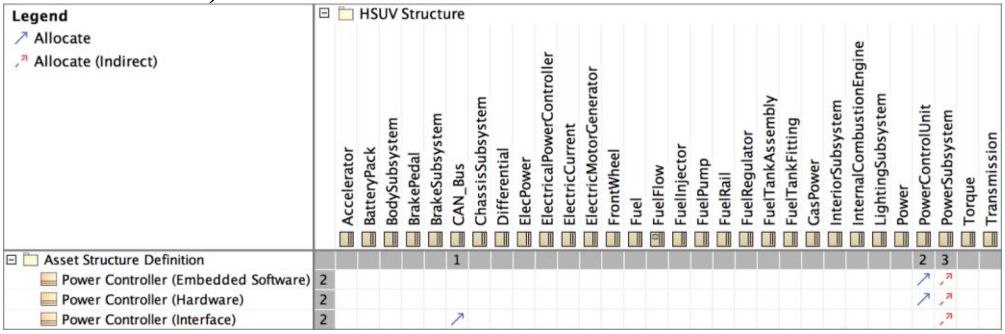
In the second phase of "Capture and Allocate Assets", the following assets that must be secured are identified: Power Control ECU Hardware, Embedded Software and its Interface. As specified in the MBSEsec implementation requirements, the assets should be created in Asset Definition Diagram (see Figure 4.18.).



Power Control Unit Asset Definition
«System Asset»
Power Controller (Interface)
«System Asset»
Power Controller (Hardware)
«Software Asset»
Power Controller (Embedded Software)

Figure 4.18. Assets for the Power Control Unit

In the follow-up step, the assets should be linked with the systems blocks with the SysML Allocate relationship. This is an interaction point between the security model and the system model, which implies that the logical system structure (blocks) should be modelled before or at the same time as assets. This traceability can be created directly in the Asset Definition Diagram or using the allocation matrix. For this case study, the allocation matrix is used (see Figure 4.19.). The matrix columns represent the system structure, and the rows represent the assets. The icon of a solid red arrow in the intersection represents direct allocation between those elements. The icon of a dashed blue arrow means that elements are indirectly allocated (e.g., if a composite block has assigned asset, the parent block has implied allocation as well).



Legend
↗ Allocate
⇢ Allocate (Indirect)

⊟ HSUV Structure

		Accelerator	BatteryPack	BodySubsystem	BrakePedal	BrakeSubsystem	CAN_Bus	ChassisSubsystem	Differential	ElecPower	ElectricalPowerController	ElectricCurrent	ElectricMotorGenerator	FrontWheel	Fuel	FuelFlow	FuelInjector	FuelPump	FuelRail	FuelRegulator	FuelTankAssembly	FuelTankFitting	GasPower	InteriorSubsystem	InternalCombustionEngine	LightingSubsystem	Power	PowerControlUnit	PowerSubsystem	Torque	Transmission
⊟ Asset Structure Definition							1																					2	3		
Power Controller (Embedded Software)	2																											↗	⇢		
Power Controller (Hardware)	2																											↗	⇢		
Power Controller (Interface)	2						↗																						⇢		

Figure 4.19. Assets allocated to HSUV blocks in the Dependency matrix

In the “Model Risks and Threats” phase, an experiment conducted by [103] is reflected. The experiment presented how a long-range wireless cyber-attack was physically tested using a real vehicle and malicious mobile application in a connected car environment. Initially, the Misuse Case diagram is used to define the high-level attack steps and the involved actors (as shown in Figure 4.20.).

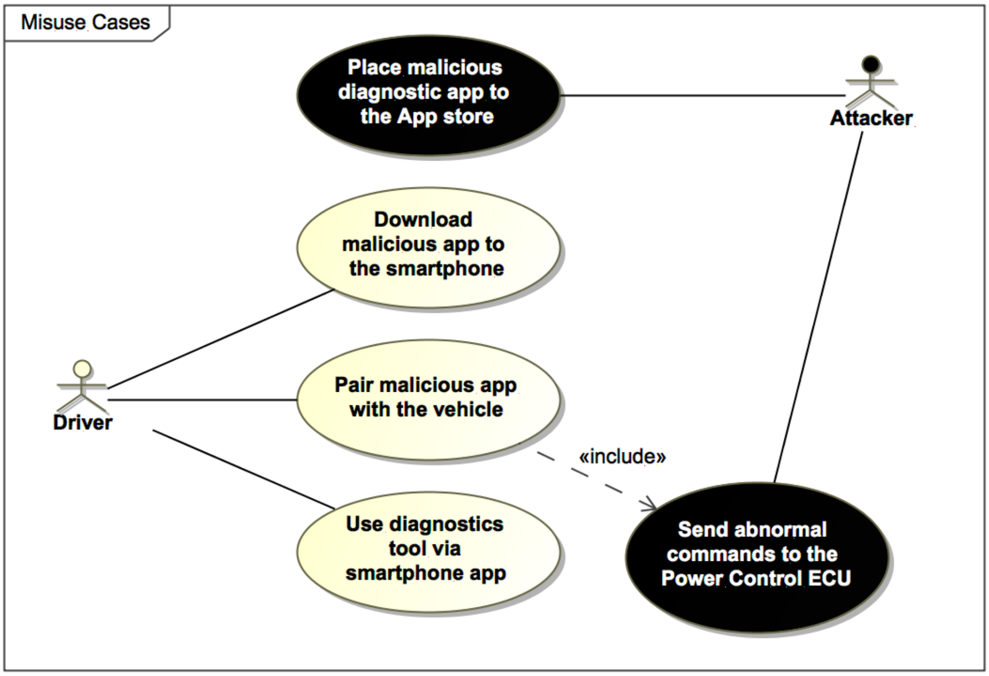

Misuse Cases
Place malicious diagnostic app to the App store
Attacker
Download malicious app to the smartphone
Driver
Pair malicious app with the vehicle
«include»
Use diagnostics tool via smartphone app
Send abnormal commands to the Power Control ECU

Figure 4.20. The Misuse Case diagram reflecting a malicious diagnostic app usage

Then, a more detailed attack scenario is modelled with the Attack/Threat scenarios diagram (an extension of SysML Activity diagram) (see Figure 4.21.). The first swimlane presents a sequence of actions performed by the malicious app, and other swimlanes represent the parts of the HSUV and what actions are invoked in each partition. As a result of this attack scenario, the vehicle has a possible fatal malfunction of *rapidly accelerate up to 200km/h* caused by the abnormal control data that was transmitted from the malicious app.

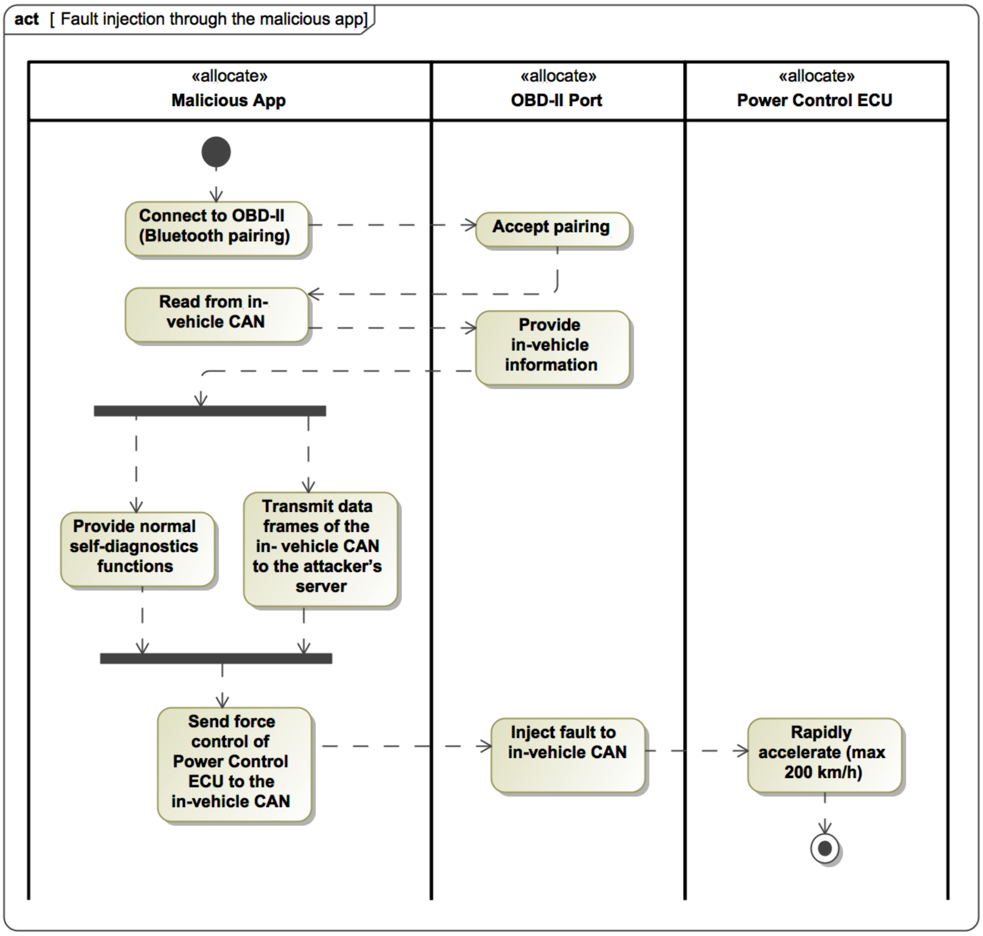
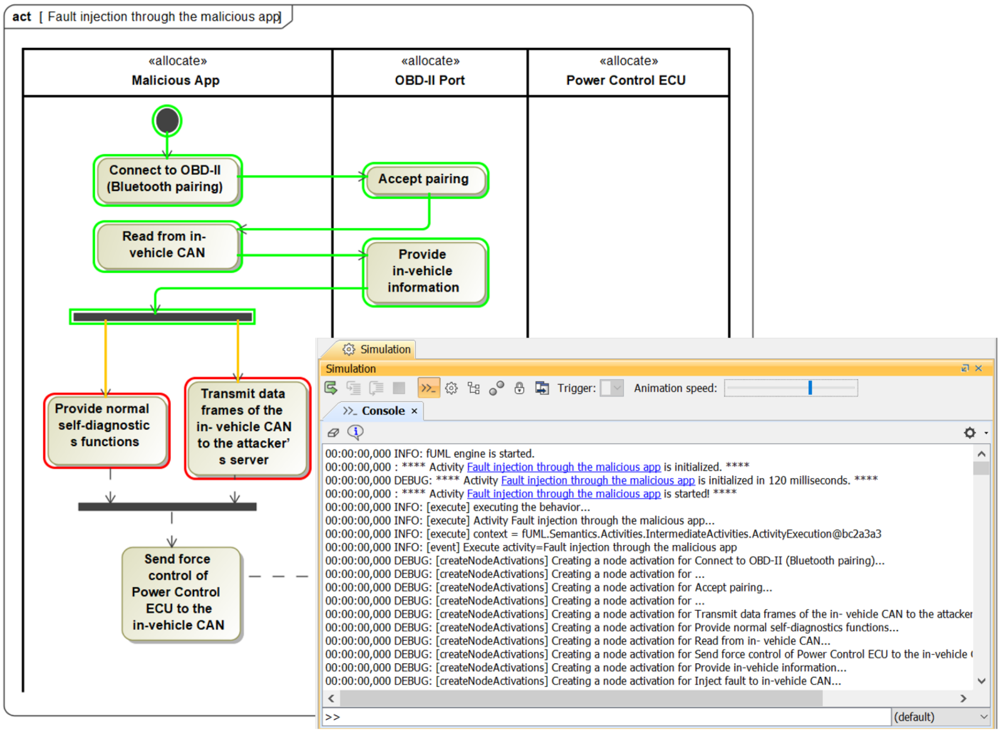

act [Fault injection through the malicious app]
«allocate» Malicious App
«allocate» OBD-II Port
«allocate» Power Control ECU
Connect to OBD-II (Bluetooth pairing)
Accept pairing
Read from in-vehicle CAN
Provide in-vehicle information
Provide normal self-diagnostics functions
Transmit data frames of the in- vehicle CAN to the attacker's server
Send force control of Power Control ECU to the in-vehicle CAN
Inject fault to in-vehicle CAN
Rapidly accelerate (max 200 km/h)

Figure 4.21. The attack scenario reflecting fault injection through a malicious app

The correctness of the "Attack/Threat scenario" diagram can be verified by running activity simulation. In Figure 4.22., it is presented how the simulation of the attack scenario is being performed with the MagicDraw 19.0 CASE tool with the Cameo Simulation Toolkit plugin. The green annotation over element indicates that the elements are already successfully executed, and the red ones indicate the active elements. The simulation console outputs the current execution state; moreover, it provides error messages if there would be any.


act [Fault injection through the malicious app]
«allocate»
Malicious App
«allocate»
OBD-II Port
«allocate»
Power Control ECU
Connect to OBD-II (Bluetooth pairing)
Accept pairing
Read from in-vehicle CAN
Provide in-vehicle information
Provide normal self-diagnostic s functions
Transmit data frames of the in- vehicle CAN to the attacker' s server
Send force control of Power Control ECU to the in-vehicle CAN
Simulation
Trigger:
Animation speed:
Console
(default)

```
00:00:00,000 INFO: fUML engine is started.
00:00:00,000 : **** Activity Fault injection through the malicious app is initialized. ****
00:00:00,000 DEBUG: **** Activity Fault injection through the malicious app is initialized in 120 milliseconds. ****
00:00:00,000 : **** Activity Fault injection through the malicious app is started! ****
00:00:00,000 INFO: [execute] executing the behavior...
00:00:00,000 INFO: [execute] Activity Fault injection through the malicious app...
00:00:00,000 INFO: [execute] context = fUML.Semantics.Activities.IntermediateActivities.ActivityExecution@bc2a3a3
00:00:00,000 INFO: [event] Execute activity=Fault injection through the malicious app
00:00:00,000 DEBUG: [createNodeActivations] Creating a node activation for Connect to OBD-II (Bluetooth pairing)...
00:00:00,000 DEBUG: [createNodeActivations] Creating a node activation for ...
00:00:00,000 DEBUG: [createNodeActivations] Creating a node activation for Accept pairing...
00:00:00,000 DEBUG: [createNodeActivations] Creating a node activation for ...
00:00:00,000 DEBUG: [createNodeActivations] Creating a node activation for Transmit data frames of the in- vehicle CAN to the attacker
00:00:00,000 DEBUG: [createNodeActivations] Creating a node activation for Provide normal self-diagnostics functions...
00:00:00,000 DEBUG: [createNodeActivations] Creating a node activation for Read from in- vehicle CAN...
00:00:00,000 DEBUG: [createNodeActivations] Creating a node activation for Send force control of Power Control ECU to the in-vehicle
00:00:00,000 DEBUG: [createNodeActivations] Creating a node activation for Provide in-vehicle information...
00:00:00,000 DEBUG: [createNodeActivations] Creating a node activation for Inject fault to in-vehicle CAN...
>>
```

Figure 4.22. The simulation of the attack scenario reflecting fault injection through a malicious app

For the final step in the "Model Risks and Threats" phase, the Threat and Risk Definition diagram with Risk, Risk Impact, Probability, Threat and Vulnerability should be modelled. Respectively, the risk of "An attacker is able to take over a Power Control ECU via the OBD-II port, reprogram it, and execute functions of Power Subsystem" is captured. This risk has the risk impact of "Lost control of HSUV acceleration" (aggregation relation between risk and risk impact). The probability of this risk occurrence is set to "Low". Based on the estimation of probability and risk impact, the "Level of Risk" is set to 5. The possible threat is "Fault injection on automotive diagnostic protocols" that potentially uses the vulnerability of "Control Area Network (CAN) protocol". The Threat and Risk definition diagram with all the relevant security elements and relations is presented in Figure 4.23.

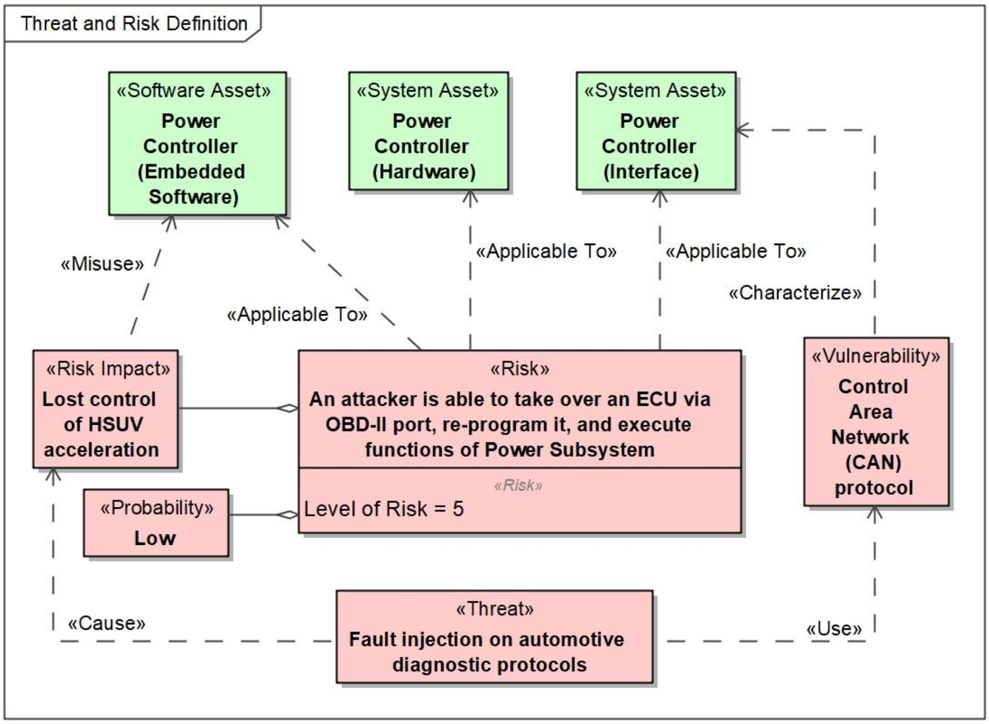

Threat and Risk Definition
«Software Asset»
Power Controller (Embedded Software)
«System Asset»
Power Controller (Hardware)
«System Asset»
Power Controller (Interface)
«Misuse»
«Applicable To»
«Applicable To»
«Applicable To»
«Characterize»
«Risk Impact»
Lost control of HSUV acceleration
«Risk»
An attacker is able to take over an ECU via OBD-II port, re-program it, and execute functions of Power Subsystem
«Risk»
Level of Risk = 5
«Vulnerability»
Control Area Network (CAN) protocol
«Probability»
Low
«Cause»
«Threat»
Fault injection on automotive diagnostic protocols
«Use»

Figure 4.23. The Threat and Risk definition diagram for the Power Control ECU

The "Decide Objectives and Controls" phase helps to identify objectives for the security controls and define the risk mitigation controls. In the HSUV case, the control objective is "System shall prevent unauthorized access to the Power Control ECU" which is associated with the corresponding security control. Moreover, the "Risk Treatment" element summarizes how the risk will be treated (by either implementing security control(s) or transferring the risk to the external party or both). In this case study, the risk will be mitigated only with the security controls (see Figure 4.24.).

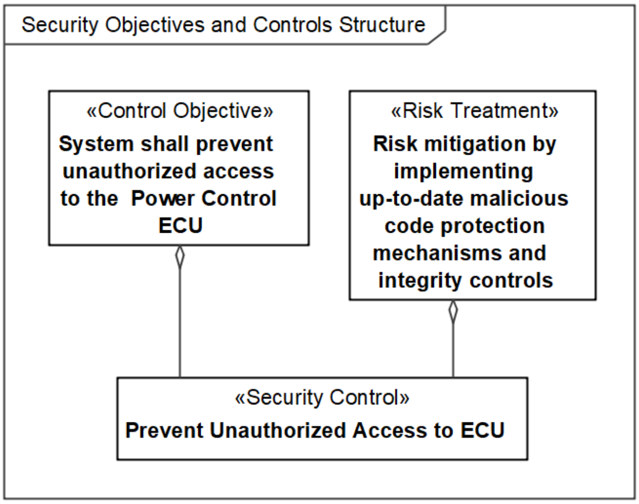

Security Objectives and Controls Structure
«Control Objective»
System shall prevent unauthorized access to the Power Control ECU
«Risk Treatment»
Risk mitigation by implementing up-to-date malicious code protection mechanisms and integrity controls
«Security Control»
Prevent Unauthorized Access to ECU

Figure 4.24. Security Objectives and Controls Structure for the Power Control Unit

The security control, in the form of an activity diagram, should present a preventive algorithm and specific actions that would allow fulfilling the security control objective. As it is shown in Figure 4.25., the activity diagram presents multilayered protection that can be reused for the different ECUs. The correctness of this diagram was verified by running fUML activity simulation. When the security control is identified and modelled, the corresponding elements (e.g., Control Objective, Security Control, Risk Treatment) could be displayed in the Threat and Risk definition diagram and validated if all the needed traceability is established.

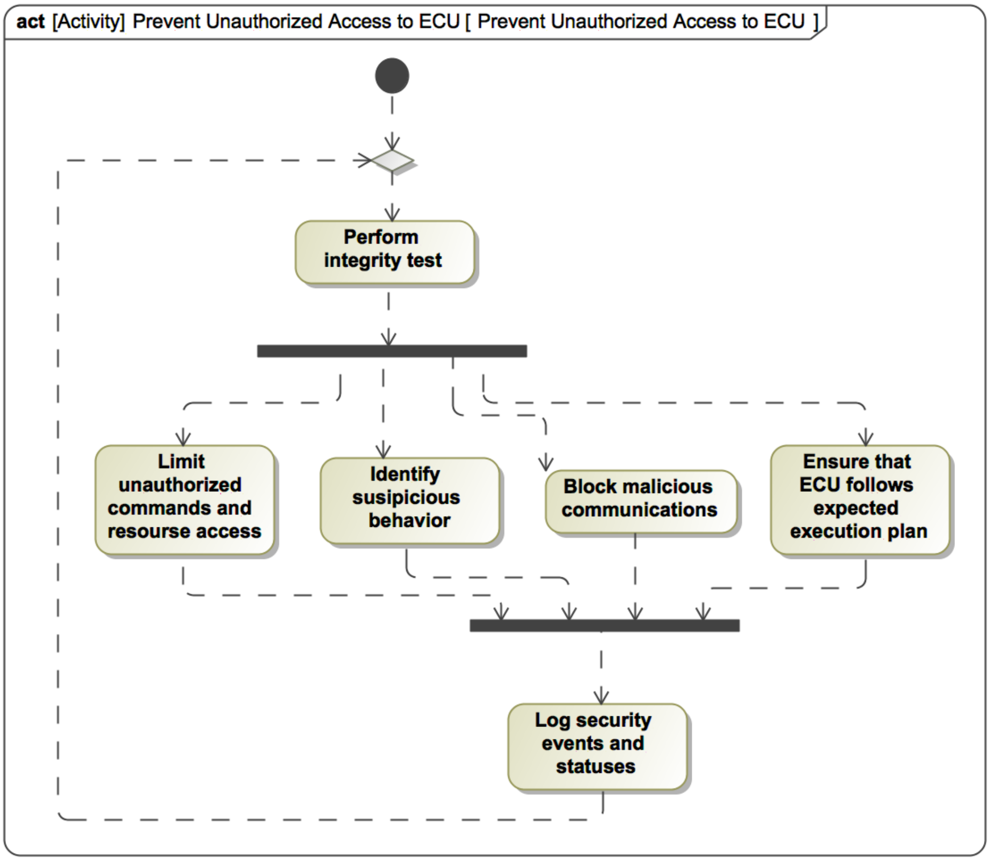
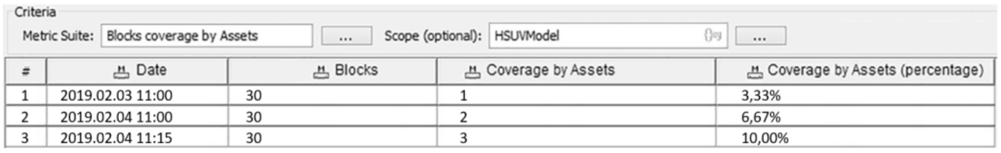

act [Activity] Prevent Unauthorized Access to ECU [Prevent Unauthorized Access to ECU]
Perform integrity test
Limit unauthorized commands and resourse access
Identify susipicious behavior
Block malicious communications
Ensure that ECU follows expected execution plan
Log security events and statuses

Figure 4.25. The security control for preventing unauthorized access to ECU

Finally, when both MBSE and security elements are created and linked in the MBSE model, the automated analysis can be executed. The OCL expression can be used as a base for the metric table (see Figure 4.26. which represents how many blocks were covered by assets in a specific time stamp), or it can be used as a query for collecting corresponding elements, or it can be used to validate the MBSE model in real time.



Criteria

Metric Suite: Blocks coverage by Assets ... Scope (optional): HSUVModel ...

#	Date	Blocks	Coverage by Assets	Coverage by Assets (percentage)
1	2019.02.03 11:00	30	1	3,33%
2	2019.02.04 11:00	30	2	6,67%
3	2019.02.04 11:15	30	3	10,00%

Figure 4.26. Metric table that presents how many blocks are covered by assets

The next automated assistance of MBSE is impact analysis. Engineers can analyse which system and security elements shall be reviewed if the initial system requirement of “Power” is changed. In Figure 4.27., there is shown the relation map diagram that presents such traceability from requirements to the system and software assets.

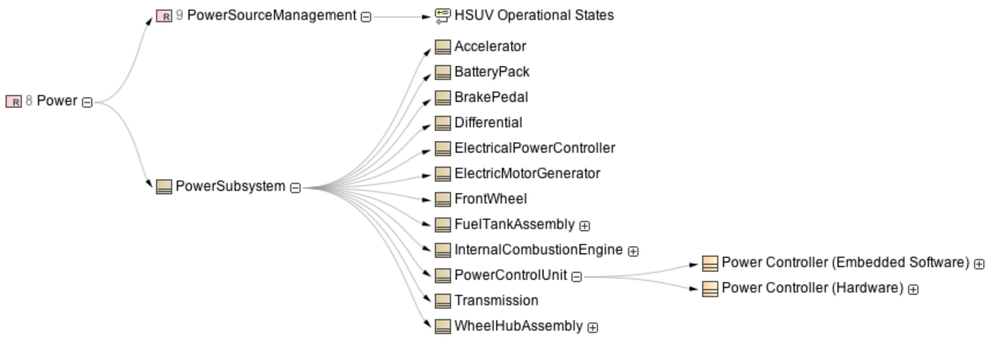


Figure 4.27. The change impact map for the "Power" requirement

4.9.1.1. Experimental evaluation results for the HSUV-Power Control ECU Case Study

This section presents the experimental evaluation results for the HSUV-Power Control ECU Case Study. First, the HSUV-Power Control ECU model statistics are provided in Table 4.5.

Table 4.5. Summary of the HSU-Power Control ECU model

Overall Model Summary	
Number of parts/subsystems of a system	6 subsystems/24 parts
Number of diagrams (not including MBSEsec diagrams)	44
Number of elements (not including MBSEsec elements)	1733
MBSEsec Summary	
Security defined for the number of parts/subsystems	0 subsystems/1 part
Number of created MBSEsec diagrams	6 MBSEsec diagrams, 2 standard UML/SysML diagrams, 3 analysis views
Number of elements (in the scope of MBSEsec packages)	160
Number of not used MBSEsec diagram types	0
Number of created MBSEsec element types	12
Number of not used MBSEsec element types	2
Number of created MBSEsec relation types	6
Number of not used MBSEsec relation types	0

The overall model summary in Table 4.5. presents the statistics for the baseline model by OMG. The HSUV system is composed of 6 subsystems and has 24 parts (as the HSUV model focuses on design decisions surrounding the power subsystem, all the parts are related to this subsystem). The number of diagrams in the model is 44, and it contains 1733 elements.

The MBSEsec summary in Table 4.5. introduces detailed information from the security perspective. As it was described in the previous section, the case study covered only one part of the Power Control ECU, which belongs to the Power subsystem. The total number of created diagrams was 11, and the MBSEsec model contained 160 elements. There was exploited all the MBSEsec specific diagrams (in total 6). The SysML Requirements diagram was used for capturing security requirements and the UML Class diagram for documenting Risk Assessment configuration. Moreover, 3 analysis views were created (i.e., allocation matrix, metric table and change impact map). There were used 12 out of 14 types of MBSEsec elements, and all the relation types (in total 6).

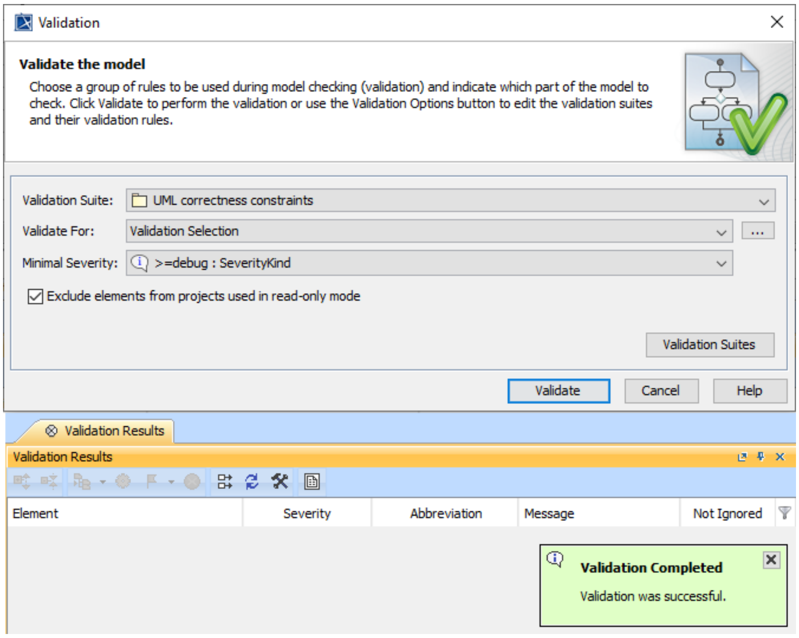
One of the method evaluation criteria was completeness. As the security phases for the HSUV were modelled with the MBSE Security Profile, which is aligned with the ISO/IEC 27001:2013 steps for establishing ISMS (the exact matching sections: *4.2.1 c) d) e) f) g)* [87]), the HSUV-Power Control ECU model proved that all the mandatory artefacts can be purposefully modelled with the MBSEsec method.

The second criteria of method conciseness looked at how many MBSEsec diagrams, elements, relationships were not used for defining the security aspect. As it is presented in Table 4.5., two types of MBSEsec elements were not necessary for this case study, one of which is “Data Asset” and the other “Third Party”. The Power Control ECU does not store data; thus, this kind of element is objectless. The “External Party” is needed to document the situation when the risk is transferred to the third party, and, for this case study, a more representative situation was showcased to present internal security control and surrounding processes.

The third criteria looked at the HSUV-Power Control ECU model correctness. The model was validated by running the UML correctness constraints validation suite, which is available with the MagicDraw 19.0 CASE tool [98]. The UML correctness constrains validation suite includes, but is not limited to, the following rules:

- A stereotype must be contained in a profile;
- The property is not owned or inherited by the context/type of nesting element;
- Property is not defined as an association end;
- If min multiplicity on the composition end is 1, no other compositions are allowed.

The validation results showed that there are no errors in any severity level (see Figure 4.28.).


Validation

Validate the model

Choose a group of rules to be used during model checking (validation) and indicate which part of the model to check. Click Validate to perform the validation or use the Validation Options button to edit the validation suites and their validation rules.

Validation Suite: UML correctness constraints

Validate For: Validation Selection

Minimal Severity: >=debug : SeverityKind

☑ Exclude elements from projects used in read-only mode

Validation Completed

Validation was successful.

Figure 4.28. The HSUV-Power Control ECU model validation results

The last criterion talked about consistency assurance between different model aspects. The MBSEsec provides traceability rules between elements in each security phase as well as between MBSEsec elements and SysML model elements. Table 4.6. details how this complete linkage was established in the HSUV-Power Control ECU model.

Table 4.6. Traceability between different aspects in the HSUV model

Source	**Target**	**Relationship type**	**Intersection**
«Software Asset» Power Controller (Embedded Software)	«Block» PowerControlUnit	Allocate	System Model <> Phase 2
«System Asset» Power Controller (Hardware)	«Block» PowerControlUnit	Allocate	System Model <> Phase 2
«System Asset» Power Controller (Interface)	«Block» PowerControlUnit	Allocate	System Model <> Phase 2
«Security Requirement» Limited signals	«Software Asset» Power Controller (Embedded Software)	Trace	Phase 1 <> Phase 2
«Security Requirement» Obfuscated access	«Software Asset» Power Controller (Embedded Software)	Trace	Phase 1 <> Phase 2
«Risk» An attacker is able to take over an ECU via OBD-II port,	«Software Asset» Power Controller (Embedded Software)	«Applicable to»	Phase 2 <> Phase 3

re-program it, and execute functions of Power Subsystem	«System Asset» Power Controller (Hardware)	«Applicable to»	Phase 2 <> Phase 3
	«System Asset» Power Controller (Interface)	«Applicable to»	Phase 2 <> Phase 3
«Risk Impact» Lost control of HSUV acceleration	«Software Asset» Power Controller (Embedded Software)	«Misuse»	Phase 2 <> Phase 3
«Vulnerability» Control Area Network (CAN) protocol	«System Asset» Power Controller (Interface)	«Characterize»	Phase 2 <> Phase 3
«Risk» An attacker is able to take over an ECU via OBD-II port, re-program it, and execute functions of Power Subsystem	«Risk Treatment» Risk mitigation by implementing up-to-date malicious code protection mechanisms and integrity controls	Aggregation	Phase 3 <> Phase 4

4.9.2. Flight Status System

The second case study presents how the MBSEsec method can be used in defining and analysing security aspects while rebuilding legacy software system. For this case study, there was selected a real-world Flight Status system, which collects data from various sources (e.g., carriers, ADS-B stations, traffic flow management systems), applies data processing and verification operations and distributes flight status information to customers. In Figure 4.29., the Flight Status System Context diagram shows how system of interest interacts with external actors. The case study for identifying security issues with MBSE while rebuilding the Flight Status software system was first time presented by Mažeika et al. in [104].

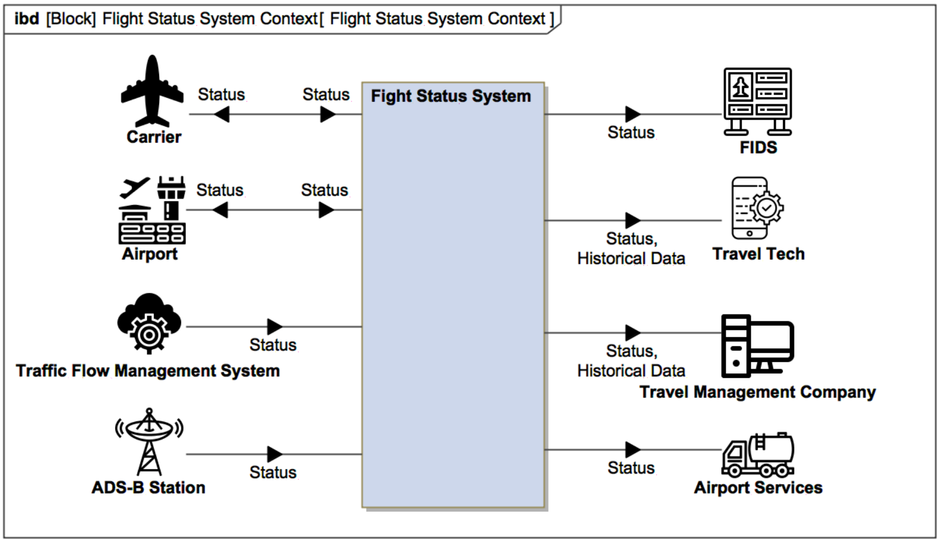


Figure 4.29. Flight Status System context diagram

Problem summary: when a decision to rebuild a legacy system has taken one of the key questions, i.e., *How to ensure that the rebuilt system is secure*? Mocanu highlights that many legacy systems were built without security in mind; moreover, the stricter regulatory requirements have helped to heighten the attention given to security for new systems [105]. This question is especially important for the Flight Status system, because any violation of availability or integrity of this system may cause disruptions and heavy losses in the aviation industry, i.e., inactivity of Paris Charles De Gaulle airport may cost over €1 million per hour to France's GDP [106].

In this case study, the re-architect method for rebuilding the legacy software system is presented, in which the legacy system is rebuilt according to the modern software development practices, i.e., using the Strangler pattern where once the new service is ready, it is put into use, and the old component is decommissioned altogether [107].

The first stage of Flight Status system modernization is the development of the "Incoming Data Processing" microservice. In such a case, for this microservice, the following security goals/requirements that reflect data integrity and system availability are created either in SysML Requirement diagram or table:

- Integrity: *the "Incoming Data Processing" microservice shall ensure the accuracy and consistency of supplier data during transfer*.
- Availability: *the "Incoming Data Processing" microservice shall ensure continuous availability to the legitimate users whenever they require it*.

The next phase is the identification of assets that must be secured. In this case study, the assets of the "Incoming Data Processing" microservice are Client Data, Flight Status Dataset and the component itself. All these elements are captured in the "Asset Definition Diagram" and allocated to the system's blocks (see Figure 4.30.).

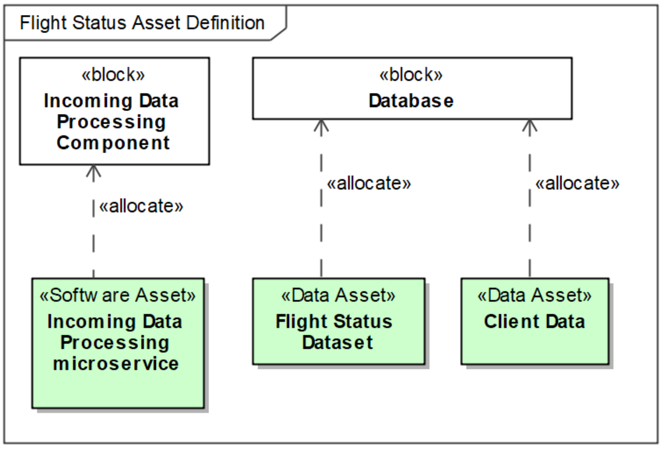


Figure 4.30. The assets of the "Incoming Data Processing" microservice and their allocations

In the "Model Risks and Threats" phase, there is presented a real-life scenario that happened for the U.S. based software development company, which sent major European airport false information on real flights, thus causing data integrity loss [106]. The high-level attack steps and involved parties (system users and hostile actors) are modelled with the Misuse Case diagram as shown in Figure 4.31.

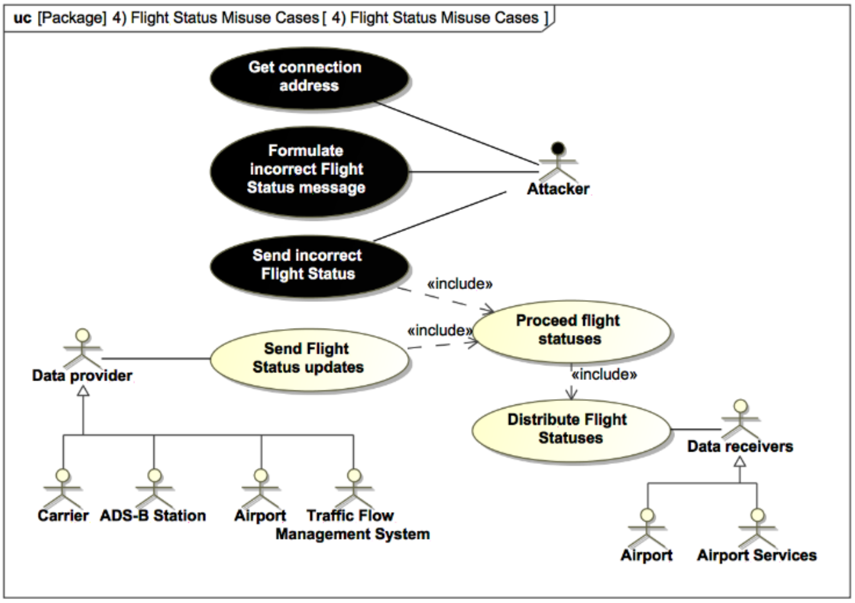

uc [Package] 4) Flight Status Misuse Cases[4) Flight Status Misuse Cases]
Get connection address
Formulate incorrect Flight Status message
Send incorrect Flight Status
Attacker
«include»
«include»
Proceed flight statuses
Send Flight Status updates
Data provider
«include»
Distribute Flight Statuses
Data receivers
Carrier
ADS-B Station
Airport
Traffic Flow Management System
Airport
Airport Services

Figure 4.31. Flight Status misuse cases

Then, a more detailed attack scenario is specified with the Attack/Threat scenarios diagram (see Figure 4.32.) and verified by running an activity simulation. The first swimlane presents a sequence of actions performed by Attacker, and other swimlanes represent the parts of the system and what actions are invoked in each partition.

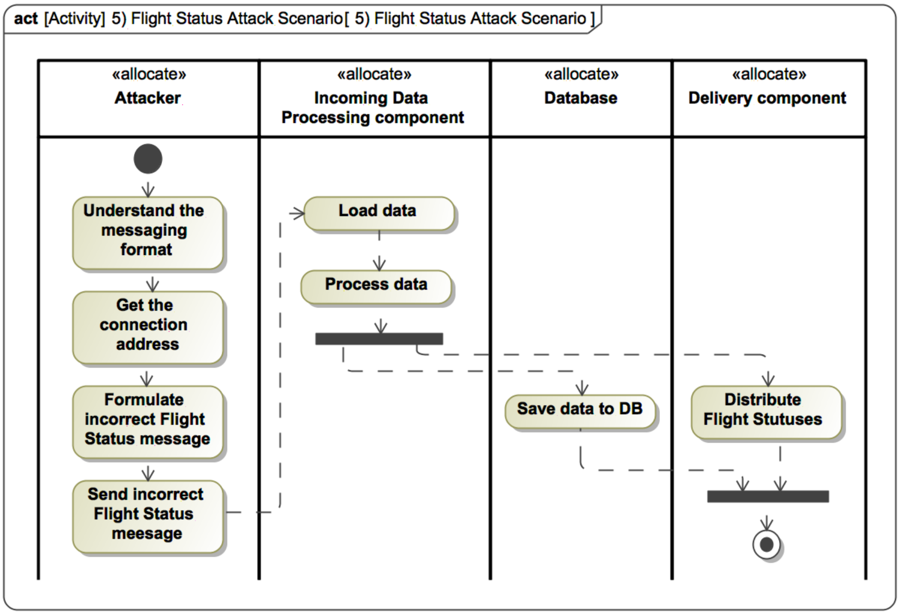

act [Activity] 5) Flight Status Attack Scenario[5) Flight Status Attack Scenario]
«allocate» Attacker
«allocate» Incoming Data Processing component
«allocate» Database
«allocate» Delivery component
Understand the messaging format
Get the connection address
Formulate incorrect Flight Status message
Send incorrect Flight Status meesage
Load data
Process data
Save data to DB
Distribute Flight Stutuses

Figure 4.32. Flight Status attack scenario

Next, the summarized information about Risk, Risk impact, Probability, Threat, Vulnerability should be provided, and possible risk treatment, security objectives and controls identified. In this case study, the possible control objective could be such: *System shall identify and authorize the sender*. The security control, in the form of the Security Process Definition diagram (an extension of SysML Activity diagram), could present how the identification and authorization mechanism (e.g., OAuth 2.0) should be implemented in the Flight Status system environment. Figure 4.33. presents all the related security elements and relations between them.

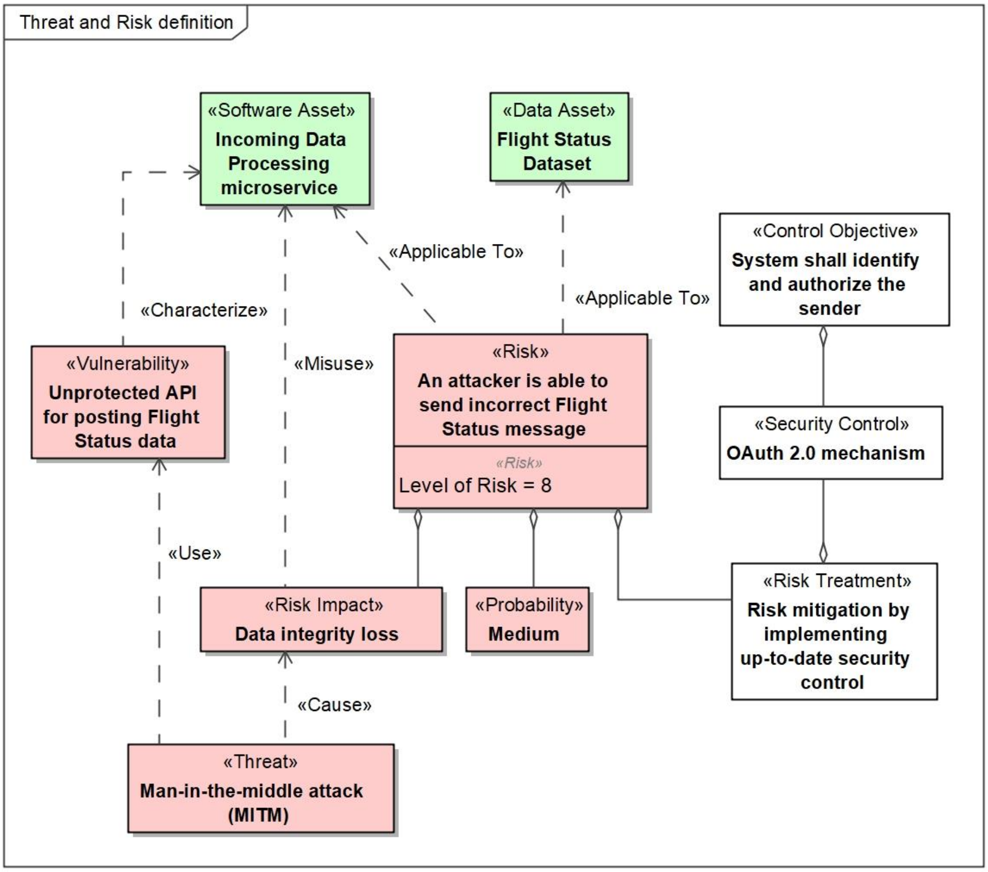
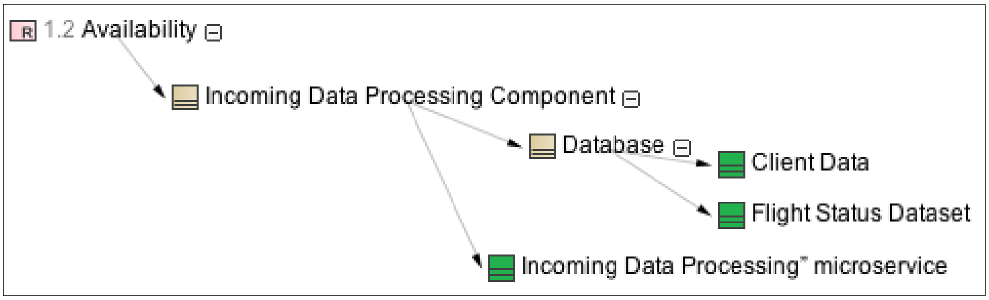

Threat and Risk definition
«Software Asset»
Incoming Data Processing microservice
«Data Asset»
Flight Status Dataset
«Applicable To»
«Applicable To»
«Control Objective»
System shall identify and authorize the sender
«Characterize»
«Misuse»
«Vulnerability»
Unprotected API for posting Flight Status data
«Risk»
An attacker is able to send incorrect Flight Status message
«Risk»
Level of Risk = 8
«Security Control»
OAuth 2.0 mechanism
«Use»
«Risk Impact»
Data integrity loss
«Probability»
Medium
«Risk Treatment»
Risk mitigation by implementing up-to-date security control
«Cause»
«Threat»
Man-in-the-middle attack (MITM)

Figure 4.33. The Threat and Risk definition diagram for the Flight Status system

When both MBSE and security artefacts are linked, the change impact can be reviewed, i.e., security engineer wants to update the *Availability* security requirement by adding a note that *the system availability shall be no less than 99.99%*. In such a case, the change impact map can show all the effected elements from the security requirement to the assets (see Figure 4.34.).



1.2 Availability
Incoming Data Processing Component
Database
Client Data
Flight Status Dataset
Incoming Data Processing" microservice

Figure 4.34. The change impact map for the "Availability" requirement

4.9.2.1. Experimental evaluation results for the Flight Status System Case Study

The summarized statistical information for the Flight Status System model study is presented in Table 4.7. (as the Flight Status system is modelled without having baseline model, the overall model summary is not provided).

Table 4.7. Summary of the Flight Status System model

MBSEsec Summary	
Security defined for the number of parts/subsystems	1 part
Number of created MBSEsec diagrams	4 MBSEsec diagrams, 1 analysis view
Number of elements (in the scope of MBSEsec packages)	179
Number of not used MBSEsec diagram types	2
Number of created MBSEsec element types	12
Number of not used MBSEsec element types	2
Number of created MBSEsec relation types	6
Number of not used MBSEsec relation types	0

For this case study, there was modelled the initial phase of the software system modernization project, in which the security aspect of the "Incoming Data Processing" microservice was created. In total, 4 MBSEsec diagrams and 1 change impact map were created. The Flight Status MBSE model contained 179 elements and covered all the artefacts from the ISO/IEC 27001:2013 standard for establishing ISMS and proved the criterion of method completeness.

In terms of method conciseness, two types of MBSEsec elements were not obligatory for this case study, one of which is the "System Asset" stereotype and the other "Third Party". The "System Asset" stereotype is applicable only to cyber-physical systems, and this case study presented a software system. Another MBSEsec stereotype that was not used was "Third Party". In the ISO 27001:2013 standard, this is one of the possible actions for the treatment of risks; thus, it needs to be in the scope of the MBSEsec method. Moreover, two MBSEsec diagram types were not created in this case study, i.e., "Security Objectives and Controls Structure" and "Security Process Definition". The "Security Process Definition" diagram for identification and authorization workflow was not separately presented (the previous case study and other examples proved that the Activity diagram is suitable for defining such algorithms). The “Security Objectives and Controls Structure” represents “Control Objectives”, “Security Controls” and “Risk Treatments”, and the same elements could be represented in the “Threat and Risk definition” diagram. In this case study, only the “Threat and Risk definition” diagram with all the elements and relations between them were modelled. In the future versions of the MBSEsec method, these diagrams may be merged in order to have a more concise method.

The UML correctness constraints validation suite was run against the Flight Status System model, and the validation results showed that there are no errors at any severity level. This proved the criterion of model correctness.

The consistency between different model aspects is presented in Table 4.8.

Table 4.8. Traceability between different aspects in the Flight Status System model

Source	**Target**	**Relationship type**	**Intersection**
«Software Asset» Incoming Data Processing microservice	«Block» Incoming Data Processing Component	Allocate	System Model <> Phase 2
« Data Asset» Flight Status Dataset	«Block» Database	Allocate	System Model <> Phase 2
«Data Asset» Client Data		Allocate	System Model <> Phase 2
«Security Requirement» Integrity	« Data Asset» Flight Status Dataset	Trace	Phase 1 <> Phase 2
«Security Requirement» Availability	«Software Asset» Incoming Data Processing microservice	Trace	Phase 1 <> Phase 2
«Risk» An attacker is able to send incorrect Flight Status message	«Software Asset» Incoming Data Processing microservice	«Applicable to»	Phase 2 <> Phase 3
	«Data Asset» Flight Status Dataset	«Applicable to»	Phase 2 <> Phase 3
«Risk Impact» Data integrity loss	«Software Asset» Incoming Data Processing microservice	«Misuse»	Phase 2 <> Phase 3
«Vulnerability» Unprotected API for posting Flight Status data	«Software Asset» Incoming Data Processing microservice	«Characterize»	Phase 2 <> Phase 3
«Risk» An attacker is able to send incorrect Flight Status message	«Risk Treatment» Risk mitigation by implementing up-to-date security control	Aggregation	Phase 3 <> Phase 4

4.10. Summary of the answers to the research questions

The summary of answers to the research questions that were introduced in the "1.3 Problem statement and research questions" are presented in Table 4.9.

Table 4.9. Summary of the research questions and answers to the research questions

Research questions	Answers to research questions
1. Is MBSE a suitable application for defining and managing security requirements and conducting security analysis for complex cyber-physical and software systems at the early stage of system creation?	Yes, the MBSE is a suitable application for these use cases because of the following reasons: • SE and the MBSE application serve as an umbrella for many disciplines that are related to designing and developing complex systems. Security requirements engineering and security analysis activities can be conducted in parallel to designing a new system. As MBSE serves as a central role and single-source-of-truth in the engineering organizations, moving security-related activities to the MBSE application would be a reasonable step forward. • The literature analysis, feasibility study and experts' survey showed that there is a need to combine these disciplines/activities.
2. Are the UML Profiles and MOF standard the right techniques and standards for creating and formalizing the domain-specific language and MBSE security method?	The UML Profiles and MOF standard is a very minimum technique for formalizing the MBSE method. The reasons for this are as follows: • The UML 2.5 Profiling capability and MOF standard gives a lightweight extension mechanism to the UML language. It allows defining stereotypes, tag definitions and constraints. • The proposed MBSE security profile was presented as a *minimum viable product* for defining security requirements and conducting security analysis in [104]. Moreover, the analysed method of UML Sec used a similar extension mechanism. In this thesis, the additional techniques and standards were selected for formalizing MBSEsec method: • DSML Definition Framework for developing and packaging custom diagrams, verification rules, samples and other customizations. • The requirements text that follows IETF RFC2119 guidelines for enabling to implement MBSEsec on any UML/SysML Case Tool.
3. How can security requirement engineering and security analysis activities be included in the MBSE process to design a secure system and leverage MBSE advantages?	In order to include security requirement engineering and security analysis activities in the MBSE process, several things are necessary (1) or recommended (2, 3, 4): 1. The MBSE security profile that contains security-related stereotypes and tag definitions. 2. The DSML definition package that contains custom diagrams, verification rules, samples and other customizations. 3. Guidelines on how to use the MBSE security

Research questions	Answers to research questions
	method, i.e., what are the prerequisites, what to accomplish in each security phase, how to establish traceability, how to run model analysis, how to structure the model. 4. Training on how to use the MBSEsec method.
4. What are the security concepts that should be introduced in systems modelling language in order to support security aspects during the early stages of system development?	There are three groups of security concepts that should be introduced in the systems modelling language: 1. Security assurance concepts, i.e., concepts that allow ensuring system security or mitigating possible risks (e.g., Security Requirement; Security Control). 2. Items to be protected (e.g., Asset). 3. Risk-related concepts that characterize hostile concepts and possible system weaknesses (e.g., Risk, Risk Impact, Vulnerability, Threat).
5. What domain specific extensions (e.g., stereotypes, diagrams, verification rules) are needed for security analysis?	The proposed MBSEsec method recommends that eleven diagrams/views should be fulfilled during security analysis (six of them are new diagram types). The total number of security-related stereotypes is twenty. Four rules for quantitative model analysis are suggested. All the needed domain specific extensions are presented in the "4.6 MBSEsec Method implementation" section.
6. Can the automated MBSE tools, including but not limited to simulation, verification and validation, change impact analysis, single source of truth, be successfully applied in the security field by using the proposed method?	Yes, the following automated MBSE tools were validated in the MBSEsec method case studies: • OCL Verification Rules; • Activity Simulation (fUML 1.1.); • Change impact analysis (what system and security elements are impacted if the initial requirement is being changed); • Metric table (e.g., how many blocks are covered by assets).
7. Does the proposed MBSE security method allow completely, concisely, correctly and consistently model security aspects of both cyber-physical and software systems in the CASE tool?	Yes, the proposed MBSE security method allow completely, concisely, correctly and consistently model security aspects of both cyber-physical and software systems in the CASE tool. Two detailed case studies were modelled to validate this: • Hybrid Sport Utility Vehicle-Power Control Unit (Cyber-physical system), • Flight Status System (Software system). Experimental evaluation presented that all the artefacts that are mandatory for defining security-related documentation (i.e., comparing with the requirements for establishing ISMS, according to the ISO/IEC 27001:2013 standard) can be created in a model-based environment. In the expert survey, the respondents suggested additional activities (e.g., information exchange analysis, CVSS vulnerability scores, threat model) that could be considered in the future version of the MBSEsec method.

5. CONCLUSIONS

The research work presented in this dissertation focused on how the security requirement engineering and security analysis activities could be integrated into the MBSE application conducted at the early phase of the system development lifecycle. The main aim of the thesis was to introduce the MBSE method for creating secure complex systems. The answers to the research objectives and main contributions are summarized below:

1. The state-of-the-art analysis of related works has revealed that MBSE is the right application for incorporating security requirements engineering and security analysis activities into the system engineering process. The security aspect is crucial in designing a complex system; however, the most popular MBSE methodologies do not provide (or provide very limitedly) such capability. Additionally, the feasibility survey, during which the representatives from ten large engineering companies from the following industries *transportation, aerospace and defence, maritime, healthcare and software,* confirmed the idea's viability of incorporating security into MBSE.
2. The baseline for the MBSE method for creating secure complex systems was a security domain model that aligned and mapped different security concepts and techniques from the security requirement engineering and security risk management fields, and UML/SysML-compliant modelling approaches for security analysis.
3. The proposed MBSEsec method consists of a UML profile with security-related stereotypes and tag definitions, the DSML definition package that contains custom diagrams, verification rules and sample projects. The UML profile and DSML definition package were prepared with the MagicDraw 19.0 CASE tool and can be used in any compatible tool. Moreover, the MBSEsec method comes with the guidelines and implementation requirements that follow IETF RFC2119 recommendations; this enables to implement MBSEsec on any UML/SysML Case Tool. The suggested method is one of the first methods in the MBSE field at the time of publication.
4. The expert evaluation, during which experts from the MBSE, engineering and academic fields were surveyed proved that the proposed MBSEsec principles and concept of integrating security analysis into MBSE are viable. The qualitative experts' evaluation showed that respondents see the importance of mitigating security risks with MBSE tools at the early stage of the system's development life cycle. All the phases from the proposed MBSEsec method are relevant and important for the experts, and the required learning time to learn additional security concepts is relatively low for practicing MBSE users. The experts noticed that additional activities or techniques (e.g., CVSS vulnerability scores, information exchange analysis, threat model) could be introduced in future versions of the MBSEsec method.
5. In the experimental evaluation part, two case studies were modelled to present how the MBSEsec method can be applied to two different real-world projects. The first one represented the Hybrid Sport Utility Vehicle, which stands for the

cyber-physical system, and the second represented how the MBSEsec method could help in rebuilding a secure Flight Status software system. The experimental evaluation showed that the MBSEsec method enables to create complete, concise, consistent and correct models.

6. REFERENCES

1. NSF-National Science Foundation, “Cyber-physical systems (CPS),” 2017. [Online]. Available: https://www.nsf.gov/pubs/2017/nsf17529/nsf17529.pdf.
2. INCOSE, “The challenge of complex systems,” [Online]. Available: http://www.incose-coa.org/the-challenge-of-complex-syst.
3. J. Guckenheimer and J. M. Ottino, “Foundations for Complex Systems Research in the Physical Sciences and Engineering,” NSF Workshop, 2008.
4. INCOSE, Systems Engineering Handbook: A Guide for System Life Cycle Processing and Activities, 4th edition, Hoboken, NJ, USA: John Wiley & Sons, 2015.
5. INCOSE, “SE vision 2025,” 2014. [Online]. Available: https://www.incose.org/AboutSE/sevision.
6. R. S. Kalawsky, J. O'Brien, S. Chong, C. Wong, H. Jia, H. Pan and P. R. Moore, “Bridging the gaps in a model-based system engineering workflow by encompassing hardware-in-the-loop simulation,” *IEEE Systems Journal,* vol. 7, no. 4, p. 593–605, 2013.
7. J. Holt, S. Perry, M. Brownsword, D. Cancila, S. Hallerstede and F. O. Hansen, “Model-based requirements engineering for system of systems,” in *roceedings of the 2012 7th International Conference on System of Systems Engineering (SoSE),* Genova, Italy, July 2012.
8. INCOSE UK, “What Is Model Based Systems Engineering (V2),” 2015. [Online]. Available: http://www.incoseonline.org.uk/Program_Files/Publications/zGuides_9.aspx?CatID=Publications.
9. É. Dubois, P. Heymans, N. Mayer and R. Matulevičius, “A Systematic Approach to Define the Domain of Information System Security Risk Management,” in *Intentional Perspectives on Information Systems Engineering,* Berlin, Springer, 2010, pp. 289-306.
10. C. Raspotnig, P. Karpati and A. L. Opdahl, “Combined Assessment of Software Safety and Security Requirements: An Industrial Evaluation of the CHASSIS Method,” *Journal of Cases on Information Technology (JCIT),* vol. 20, no. 1, pp. 46-69, 2018.
11. G. Styles and R. S. Kalawsky, “Research Top Challenges for MBSE in Industry 4.0 and IoT – Workshop Report,” 2015.
12. P. H. Nguyen, S. Ali and T. Yue, “Model-based security engineering for cyber-physical systems: a systematic mapping study,” *Information and Software Technology,* vol. 83, pp. 116-135, 2017.
13. B. L. Papke, “Enabling design of agile security in the IOT with MBSE,” in *Proceedings of the 2017 12th System of Systems Engineering Conference (SoSE),* Waikoloa, HI, USA, 2017.
14. J. Jürjens and P. Shabalin, “Tools for secure systems development with UML,”

International Journal on Software Tools for Technology Transfer, vol. 9, pp. 527-544, 2007.

15. A. M. Madni and S. Purohit, “Economic Analysis of Model-Based Systems Engineering,” *Systems* , vol. 7, no. 1, pp. 1-12, 2019.
16. E. Carroll, “Systematic Literature Review: How Is Model-Based Systems Engineering Justified?,” 2016. [Online]. Available: http://www.omgwiki.org/MBSE/lib/exe/fetch.php?media=mbse:incose_mbse_iw_2017:sand2016-11485_pe_uur_161109_howismodel-basedsystemsengineeringjustified_.pdf.
17. NIST, “Security Considerations in the System Development Life Cycle,” 2008. [Online]. Available: https://www.nist.gov/publications/security-considerations-system-development-life-cycle.
18. A. Morkevicius, A. Aleksandraviciene, D. Mazeika, L. Bisikirskiene and Z. Strolia, “MBSE Grid: A Simplified SysML-Based Approach for Modeling Complex Systems,” in *INCOSE International Symposium,* Adelaide, Australia, 2017.
19. A. R. Hevner, S. T. March, J. Park and S. Ram, “Design Science in Information Systems Research,” *MIS Quarterly,* vol. 28, no. 1, pp. 75-105, 2004.
20. T. W. Olle, H. G. Sol and I. G. MacDonald, Information Systems Methodologies: A Framework for Understanding, Boston, MA, USA: Addison-Wesley, 1991.
21. KTU, “Research Report,” 2018. [Online]. Available: https://ktu.edu/wp-content/uploads/2016/08/Moksliniai_tyrimai_2018_ataskaita.pdf.
22. Microsoft, “Microsoft Security Development Lifecycle,” 2018. [Online]. Available: https://www.microsoft.com/en-us/securityengineering/sdl.
23. Microsoft, “Security Development Lifecycle for Agile Development,” 2012. [Online]. Available: https://docs.microsoft.com/en-us/previous-versions/windows/desktop/ee790621(v=msdn.10).
24. A. Kilčiauskas, G. Butkus and E. Sakalauskas, “Authenticated Key Agreement Protocol Based on Provable Secure Cryptographic Functions,” *INFORMATICA,* vol. 31, no. 2, p. 277–298, 2020.
25. A. Kajackas, R. Rainys and A. Aputis, “Assessment of Cyber Attacks Influence over Internet Network,” *Elektronika Ir Elektrotechnika,* vol. 113, no. 7, pp. 89-92, 2011.
26. J. Janulevičius, “Method of information security risk analysis for virtualized systems. Doctoral dissertation,” Vilnius Gediminas Technical University, Vilnius, 2016.
27. Sebok, “Systems Engineering: Historic and Future Challenges,” 2019. [Online]. Available: https://www.sebokwiki.org/wiki/Systems_Engineering:_Historic_and_Future_Challenges.
28. IEEE Reliability Society, “Systems of Systems,” 2014. [Online]. Available:

https://rs.ieee.org/technical-activities/technical-committees/systems-of-systems.html.

29. International Organization for Standardization, “ISO/IEC/IEEE 15288:2015 Systems and software engineering — System life cycle processes,” 2015.
30. INCOSE Canada, “Modern Challenges in Systems Engineering,” 2015. [Online]. Available: http://incosecanada.weebly.com/conference-2015.html.
31. I. Graessler, J. Hentze and T. Bruckmann, “V-Models for Interdisciplinary Systems Engineering,” in *DESIGN 2018 15th International Design Conference*, Dubrovnik, Croatia, 2018.
32. Federal-Highway-Administration, “Systems Engineering for Intelligent Transport Systems. US-DoT, US Department of Transportation.,” 2007.
33. L. E. Hart, “Introduction To Model-Based System Engineering (MBSE) and SysML,” [Online]. Available: https://www.incose.org/docs/default-source/delaware-valley/mbse-overview-incose-30-july-2015.pdf.
34. J. A. Estefan, “INCOSE Survey of MBSE Methodologies,” INCOSE TD 2007-003-02, Seattle, WA, USA, 2008.
35. INCOSE Technical Operations, “INCOSE SE Vision 2020,” 2007. [Online].
36. C. Delp, D. Lam, E. Fosse and C.-Y. Lee, “Model based document and report generation for systems engineering,” in *IEEE Aerospace Conference*, Big Sky, MT, USA, 2013.
37. OMG, “Unified Modeling Language,” 2017. [Online]. Available: https://www.omg.org/spec/UML/About-UML/.
38. OMG, “Model Driven Architecture,” [Online]. Available: https://www.omg.org/mda/.
39. D. Silingas and R. Butleris, “Towards customizing UML tools for enterprise architecture modeling,” in *IADIS international conference*, Barcelona, Spain, 2009.
40. A. Morkevicius and S. Gudas, “Enterprise Knowledge Based Software Requirements Elicitation,” *Information Technology and Control*, vol. 40, no. 3, pp. 181-190, 2011.
41. Object Management Group, “About the OMG System Modeling Language specification version 1.6,” [Online]. Available: https://www.omg.org/spec/SysML/About-SysML/.
42. Object Management Group, “What is SysML?,” 2017. [Online]. Available: https://www.omgsysml.org/what-is-sysml.htm.
43. R. Cloutier and M. Bone, “Compilation of SysML RFI- Final Report, Systems Modeling Language (SysML),” 2010. [Online]. Available: http://www.omgwiki.org/OMGSysML/lib/exe/fetch.php?media=sysml-roadmap:omg_rfi_final_report_02_20_2010-1.pdf.
44. S. C. Spangelo, D. Kaslow, C. Delp, B. Cole, L. Anderson, E. Fosse and B. Sam, “Applying Model Based Systems Engineering (MBSE) to a standard

CubeSat," in *IEEE Aerospace Conference*, Big Sky, MT, USA, 2012.

45. INCOSE, "MBSE Wiki: Mehtodology and Metrics," 2020. [Online]. Available: http://www.omgwiki.org/MBSE/doku.php?id=mbse:methodology.
46. D. Mazeika, A. Morkevicius and A. Aleksandraviciene, "MBSE driven approach for defining problem domain," in *System of Systems Engineering (SoSE)*, Kongsberg, Norway, 2016.
47. INCOSE, "Object-Oriented SE Method," [Online]. Available: https://www.incose.org/incose-member-resources/working-groups/transformational/object-oriented-se-method.
48. P. Pearce and M. Hause, "ISO-15288, OOSEM and Model-Based Submarine Design," in *SETE/APCOSE*, Brisbane, Australia, 2012.
49. H.-P. Hoffmann, Model-Based Systems Engineering with Rational Rhapsody and Rational Harmony for Systems Engineering - Deskbook 4, 2013.
50. B. P. Douglass, Harmony aMBSE Deskbook (Version 1.00), 2017.
51. B. P. Douglass, "Agile Model-Based Systems Engineering (aMBSE)," in *IBM Symposium Systemes*, Paris, 2014.
52. T. Weilkiens, SYSMOD - The Systems Modeling Toolbox - Pragmatic MBSE with SysML, MBSE4U - Tim Weilkiens, 2015.
53. M. D. Ingham, R. D.Rasmussen, M. B. Bennett and A. C. Moncada, "Generating requirements for complex embedded systems using State Analysis," *Acta Astronautica,* vol. 58, no. 12, pp. 648-661, 2006.
54. D. A. Wagner, M. B. Bennett, R. Karban, N. Rouquette, S. Jenkins and M. Ingham, "An ontology for State Analysis: Formalizing the mapping to SysML," in *IEEE Aerospace Conference*, Big Sky, MT, USA, 2012.
55. D. Long and Z. Scott, A Primer for Model-Based Systems Engineering, 2012.
56. D. Mellado, C. Blanco, L. E.Sánchez and E. Fernández-Medina, "A systematic review of security requirements engineering," *Computer Standards & Interfaces,* vol. 32, no. 4, p. 153–165, 2010.
57. US-CERT, "SQUARE process," 2013. [Online]. Available: https://www.us-cert.gov/bsi/articles/best-practices/requirements-engineering/square-process.
58. J. Viega, "Building security requirements with CLASP," in *SESS@ICSE*, St. Louis, Missouri, USA, 2005.
59. D. Mellado, E. Fernández-Medina and M. Piattini, "Applying a Security Requirements Engineering Process," in *European Symposium on Research in Computer Security (ESORICS'06)*, Guildford, UK, 2006.
60. CORAS, "The CORAS method," 2015. [Online]. Available: http://coras.sourceforge.net/.
61. B. Fabian, S. Gurses, M. Heisel, T. Santen and H. Schmidt, "A comparison of security requirements engineering methods," *Requirements Engineering,* vol. 15, no. 1, p. 7–40, 2010.
62. P. Salini and S. Kanman, "Survey and analysis on security requirements

engineering," *Computers & Electrical Engineering,* vol. 38, no. 6, p. 1785–1797, 2012.

63. ISO/IEC 13335-1:2004, Information technology - Security techniques - Management of information and communications technology security: Part 1, International Organization for Standardization, 2007.
64. E. Albrechtsen, "Safety vs Security," 2003. [Online]. Available: http://www.iot.ntnu.no/users/albrecht/rapporter/notat%20safety%20v%20security.pdf.
65. S. Kriaa, L. Pietre-Cambacedes, M. Bouissou and Y. Halgand, "A survey of approaches combining safety and security for industrial control systems," *Reliability Engineering & System Safety,* vol. 139, p. 156–178, 2015.
66. S. Kriaa, L. Pietre-Cambacedes, M. Bouissou and Y. Halgand, "A survey of approaches combining safety and security for industrial control systems," *Reliability Engineering & System Safety,* vol. 139, pp. 156-178, 2015.
67. Object Management Group, "About the Unified Architecture Framework Specification Version 1.0," 2017. [Online]. Available: http://www.omg.org/spec/UAF/1.0/Beta2/.
68. A. Morkevicius, L. Bisikirskiene and G. Bleakley, "Using a systems of systems modeling approach for developing Industrial Internet of Things applications," in *12th System of Systems Engineering Conference (SoSE),* Waikoloa, HI, USA, 2017.
69. P. Vaughan, "Integrating UPDM with SysML and UML on a DoD acquisition program," in *OMG UAF & MBSE information day,* Reston, VA, USA, 2015.
70. G. J. Bleakley and M. Haus, "The united architecture framework the Internet of Things and power systems," 2016. [Online]. Available: http://www.omg.org/news/meetings/tc/ca-16/special-events/iot-presentations/Hause-Bleakley.pdf.
71. SANS Institute, "Using the department of defense architecture framework to develop security requirements," 2014. [Online]. Available: https://www.sans.org/reading-room/whitepapers/bestprac/department-defense-architecture-framework-develop-security-requirements-34500.
72. NIST, "NIST, DOD, intelligence agencies join forces to secure U.S. cyber infrastructure," 2009. [Online]. Available: https://www.nist.gov/news-events/news/2009/06/nist-dod-intelligence-agencies-join-forces-secure-us-cyber-infrastructure.
73. C. Raspotnig, V. Katta, P. Karpati and A. L. Opdahl, "Enhancing CHASSIS: A Method for Combining Safety and Security," in *International Conference on Availability, Reliability and Security,* Regensburg, Germany, 2013.
74. C. Raspotnig, P. Karpati and V. Katta, "A Combined Process for Elicitation and Analysis of Safety and Security Requirements," in *Enterprise, Business-Process and Information Systems Modeling,* Heidelberg, Germany, 2012.
75. Y. Roudier and L. Apvrille, "SysML-Sec: a model driven approach for

designing safe and secure systems," in *3rd International Conference on Model-Driven Engineering and Software Development*, Angers, France, 2015.

76. Y. Roudier and L. Apvrille, "SysML-sec: a model-driven environment for developing secure embedded systems," in *8th conference on the security of network architecture and information systems (SARSSI'2013)*, Mont de Marsan, France, 2013.
77. J. Jürjens, "UMLsec: extending UML for secure systems development,," in *The Unified Modeling Language*, 2002.
78. H. Mouratidis and J. Jurjens, "From goal-driven security requirements engineering to secure design," *International Journal of Intelligent Systems*, vol. 25, no. 8, p. 813–840, 2010.
79. F. Braber, T. Dimitrakos, B. A. Gran, M. S. Lund, K. Stølen and J. Aagedal, "The CORAS methodology: model-based risk management using UML and UP," in *UML and the Unified Process*, Buenos Aires, Argentina, 2003.
80. D. Watson and A. Jones, "Risk Management," in *Digital Forensics Processing and Procedures*, 2013, pp. 109-176.
81. R. Evans, A. Tsohou, T. Tryfonas and T. Morgan, "Engineering secure systems with ISO 26702 and 27001," in *5th International Conference on System of Systems Engineering*, Loughborough, UK, 2010.
82. E. Aroms, "NIST Special Publication 800-30 Risk Management Guide for Information Technology Systems," Create Space, Scotts Valley, 2012.
83. International Standards Organisations, "ISO/IEC 27005:2018 Information technology — Security techniques — Information security risk management," Geneva, Switzerland, 2018.
84. International Standards Organisations, "ISO 31000:2018 Risk management — Guidelines," Geneva, Switzerland, 2018.
85. C. J. Alberts, S. Behrens, R. D. Pethia and W. R. Wilson, "Operationally Critical Threat, Asset, and Vulnerability Evaluation (OCTAVE) Framework, Version 1.0," Software Engineering Institute, 1999 .
86. ISMS.online, "Information Security Management System (ISMS)," [Online]. Available: https://www.isms.online/information-security-management-system-isms/.
87. International Standards Organisation, "ISO/IEC 27001 - Information Technology—Security Techniques—Information Security Management Systems—Requirements," Geneva, Switzerland, 2013.
88. D. Mažeika and R. Butleris, "Integrating security requirements engineering into MBSE: Profile and guidelines," *Security and Communication Networks*, p. 1–12, 2020.
89. Object Management Group, "About the Meta Object Facility specification version 2.5.1," 2016. [Online]. Available: https://www.omg.org/spec/MOF;jsessionid=FE501A2F1AABFF3587B96AA1DE7F4EFF.

90. UML diagrams, “UML, Meta Meta Models and Profiles,” 2020. [Online]. Available: https://www.uml-diagrams.org/uml-meta-models.html.
91. D. Šilingas, R. Vitiutinas, A. Armonas and L. Nemuraitė, “Domain-specific modeling environment based on UML profiles,” in *Information technologies,* Kaunas, Lithuania, 2009.
92. S. Bradner, “Key words for use in RFCs to Indicate Requirement Levels,” 1997. [Online]. Available: https://tools.ietf.org/html/rfc2119.
93. CATIA No Magic, “Distributing Resources,” [Online]. Available: https://docs.nomagic.com/display/MD190/Distributing+Resources.
94. D. Mažeika and R. Butleris, “MBSEsec: Model-Based Systems Engineering Method for Creating Secure Systems,” *Applied Sciences,* vol. 10, no. 7, pp. 1-18, 2020.
95. R. Libby and R. K. Blashfield, “Performance of a composite as a function of the number of judges,” *Organizational Behavior and Human Performance,* vol. 21, no. 2, pp. 121-129, 1978.
96. Laerd Statistics, “Kruskal-Wallis H Test using Stata,” 2014. [Online]. Available: https://statistics.laerd.com/stata-tutorials/kruskal-wallis-h-test-using-stata.php.
97. Stata, “Stata: Software for Statistics and Data Science,” 2020. [Online]. Available: https://www.stata.com/.
98. CATIA No Magic, “Predefined validation suites,” 2020. [Online]. Available: https://docs.nomagic.com/display/MD190/Predefined+validation+suites.
99. A. Chattopadhyay and K.-Y. Lam, “Security of autonomous vehicle as a cyber-physical system,” in *7th International Symposium on Embedded Computing and System Design (ISED),* Durgapur, India, 2017.
100. Intel , “Automotive Security Research Workshops,” 2015. [Online]. Available: https://www.intel.com/content/dam/www/public/us/en/documents/product-briefs/automotive-security-research-workshops-summary.pdf.
101. ISO/IEC, “Information technology - Guidelines for the management of IT Security - Part 3,” [Online]. Available: https://www.sis.se/api/document/preview/611327/.
102. M. Riaz and L. Williams, “Security requirements patterns: Understanding the science behind the art of pattern writing,” in *2012 Second IEEE International Workshop on Requirements Patterns (RePa),* Chicago, IL, USA, 2012.
103. S. Woo, H. J. Jo and D. H. Lee, “A practical wireless attack on the connected car and security protocol for in-vehicle CAN,” *IEEE Transactions on Intelligent Transportation Systems,* vol. 16, no. 2, pp. 993 - 1006, 2015.
104. D. Mažeika and R. Butleris, “Identifying Security Issues with MBSE while Rebuilding Legacy Software Systems,” in *IEEE 15th International Conference on System of Systems Engineering (SoSE),* Budapest, Hungary,

2020.

105. V. Mocanu, “Requirements for Security Enhancements to Legacy Software with RUP,” *Information Security Journal: A Global Perspective,* vol. 19, no. 4, pp. 226-236, 2010.

106. SESAR, “Addressing airport cyber-security. Final report,” 2016. [Online]. Available: https://www.sesarju.eu/sites/default/files/documents/news/Addressing_airport_cyber-security_Full_0.pdf.

107. S. Behara, “Monolith to Microservices Using the Strangler Pattern,” 2018. [Online]. Available: https://dzone.com/articles/monolith-to-microservices-using-the-strangler-patt.

7. LIST OF PUBLICATIONS OF DONATAS MAŽEIKA ON THE THEME OF DISSERTATION

Articles in Journals referred in Clarivate Analytics Web of Science (CA WoS) database:

1. D. Mažeika, R. Butleris. **Integrating Security Requirements Engineering into MBSE: Profile and Guidelines** // Security and Communication Networks Journal (Hindawi/Wiley). London, UK, 2020; pp. 1–12.
2. D. Mažeika; R. Butleris. **MBSEsec: Model-Based Systems Engineering Method for Creating Secure Systems** // Applied Sciences journal; MDPI: Basel, Switzerland, 2020; pp. 1-18.

Conference articles:

1. D. Mažeika; A. Morkevičius, A. Aleksandravičienė. System of Systems Engineering Conference (SoSE) 2016 11th, Kongsberg, Norway. "**MBSE driven approach for defining problem domain**", Date of Conference: 12-16 June 2016, Date Added to IEEE Xplore: 15 August 2016, DOI: 10.1109/SYSOSE.2016.7542911, Publisher: IEEE
2. A. Morkevičius, A. Aleksandravičienė, D. Mažeika, L. Bisikirskienė, Ž. Strolia. 27th Annual INCOSE International Symposium, Adelaide, Australia **MBSE Grid: A Simplified SysML-Based Approach for Modeling Complex Systems**. INCOSE INTERNATIONAL SYMPOSIUM Volume 27, Issue 1, July 2017, Pages: 136–150, DOI: 10.1002/j.2334-5837.2017.00350.x
3. D. Mažeika; R. Butleris. 9th International Workshop "Data analysis methods for software systems" – DAMSS 2017, Druskininkai, Lithuania. **Model-Based Systems Engineering Approach for Creating Secure Complex Systems**. Proceeding of 9th International Workshop DAMSS 2017: Druskininkai, Lithuania, November 30 - December 3, 2017 ISBN: 978-9986-680-64-2
4. D. Mažeika; R. Butleris. IEEE 15th International Conference of System of Systems Engineering (SoSE), 2020, Budapest, Hungary. "**Identifying Security Issues with MBSE while Rebuilding Legacy Software Systems**". Date of Conference: 2-4 June 2020, Date Added to IEEE Xplore: 1 July 2020, DOI: 10.1109/SoSE50414.2020.9130491, Publisher: IEEE

SL344. 2021-05-04, 12,25 leidyb. apsk. l. Tiražas 14 egz. Užsakymas 117.
Išleido Kauno technologijos universitetas, K. Donelaičio g. 73, 44249 Kaunas
Spausdino leidyklos „Technologija“ spaustuvė, Studentų g. 54, 51424 Kaunas