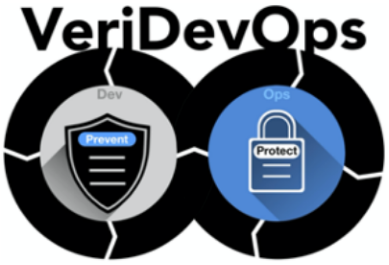

VeriDevOps

Automated Protection and Prevention to Meet Security

Requirements in DevOps Environments

D1.4 VeriDevOps Framework Architecture and Roadmap

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 957212. This document reflects only the author's view and the European Commission is not responsible for any use that may be made of the information it contains.

Contract number:	957212
Project acronym:	VeriDevOps
Project title:	VeriDevOps: Automated Protection and Prevention to Meet Security Requirements in DevOps Environments
DELIVERY DATE	
Author(s):	Main Editor: Andrey Sadovykh (Softeam)
Partners contributed:	SOFT, ABO, MI, IKER, MDH
Date:	22.09.2021
Version	1
Revision:	4
Abstract:	This is a live document that outlines the architecture of the VeriDevOps framework and its constituent parts. The document also summarised the requirements analysis by providing traceability of the case study requirements and tool feature requirements. The document provides a roadmap as a list of tool features to be available at various milestones related to the case study development and evaluation. The document is generated out of a shared architecture model that partners jointly maintain throughout the project.
Status:	

DOCUMENT REVISION LOG

VERSION	REVISION	DATE	DESCRIPTION	AUTHOR
1.0	01		Final Draft	Andrey Sadovykh, Softeam
1.0	02		Reviewed	Andrey Sadovykh, Softeam
1.0	03		Updated	Andrey Sadovykh, Softeam
1.0	04		Resubmitted	Andrey Sadovykh, Softeam

TABLE OF CONTENTS

Executive Summary

This is a live document that outlines the architecture of the VeriDevOps framework and its constituent parts. The document also summarised the requirements analysis by providing traceability of the case study requirements and tool feature requirements. The document provides a roadmap as a list of tool features to be available at various milestones related to the case study development and evaluation.
The document is generated out of a shared architecture model that partners jointly maintain throughout the project.

1 Framework Architecture

This deliverable shows a snapshot of the evolving VDO architecture maintained by partners as a shared UML model. In this live document, we describe the VeriDevOps Framework and its constituent parts.
The description is done by following a common pattern:
- we describe the high-level purpose of each components including possible roadmap for feature implementation;
- we outline the functional interfaces that help to figure out the main features and possible means for integration;
- we shortly detail the subordinates – the constituent parts of each component;
- for the individual tools we clarify their relation to the VeriDevOps Framework.
The document also highlights the traceability between Case study requirements, the conceptual tools of the framework and the tools of the individual partners.

More on the architecture specification approach can be found in:
Sadovykh A., Truscan D., Bruneliere H. Applying Model-based Requirements Engineering in Three Large European Collaborative Projects: An Experience Report. Requirements Engineering Conference – 2021.

The VeriDevOps Framework is the main technical result of the project as described in the project proposal. The Framework groups several interconnected tool sets for Security Requirements Generation, Reactive Protection at Runtime as well as for Prevention at Design and Development. Those tool sets are highly interconnected to achieve the goal of linking security requirements with design analysis, verification at code level and the runtime

analysis of systems.

The VeriDevOps tool sets mix concrete tool components provided and developed by VeriDevOps partners. These concrete tools differ in licensing policies and maturity. While there are many mature commercial and open-source tools, others are more experimental. These tools should implement the interfaces and features of the VeriDevOps Framework and should be interchangeable to a certain extent. The case studies will employ a mix of those tools that better correspond to the requirements and will fit better to the methodology or an industry practice.

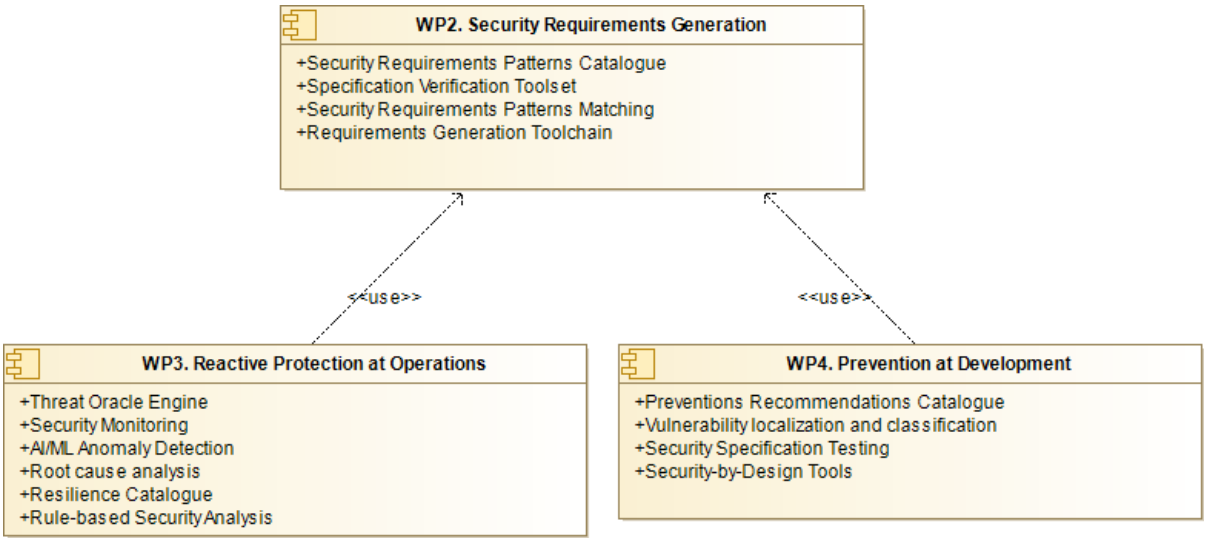

Figure 1 Framework Architecture Overview

The Framework Architecture consists of several components related to the work packages of the VeriDevOps project. These components are further described below.

1.1 WP2. Security Requirements Generation

As the proposal stated, security requirements will play a central role in VeriDevOps and their availability and quality are of utmost importance. In this project, we aim at extracting security requirements from different publicly available vulnerability catalogues, as well as from security technical implementation guide (i.e. STIG), NIST and security standards such as IEC 26443. For this purpose, we will use natural language processing (NLP) to automatically convert textual requirements into formal specifications that will be used later for generating different artefacts. For security requirements created by industrial engineers, we will use a front-end tool for requirements specification and complemented it by applying a pattern-based method to automatically generate formalized requirements.

Thus, the toolset provides automation in extraction, formalization and verification of the security requirements from users' requirements, vulnerability databases, security agencies,

security implementation guides, and standards. The extracted requirements will be used both for artefact generation and for verification of formal design specifications of the system.

1.1.1 Related Case Study Requirements for WP2. Security Requirements Generation

Properties	Definition
Criticality: High Release: Intermediate-M24	FAG-010: During software development, VeriDevOps outcome will help in testing patches before deploying them to the entire network, then reducing time and effort.
Criticality: High Release: Intermediate-M24	FAG-020: During machine operation, VeriDevOps will produce automatic detection of known vulnerabilities and attacks, analysis of their root cause and providing effective automatic countermeasures (reaction), or manual recommendations for decision support.
Criticality: High Release: Intermediate-M24	FAG-030: FAGOR needs an automatic procedure capable of interpreting these security requirements or recommendations such as NIST or STIG, and finding when the device is implementing them and rising alerts when devices do not fulfil them.
Criticality: High Release: Intermediate-M24	FAG-040: FAGOR needs to add to its devices a capability which can be automatic search for vulnerabilities.
Criticality: High Release: Intermediate-M24	FAG-050: FAGOR expects automated requirement interpretation and implementation (WP2)
Criticality: High Release: Intermediate-M24	FAG-110: VDO Framework shall provide means to interpret the NIST framework requirements and check in the system if the requirements are implemented or not.
Criticality: High Release: Intermediate-M24	FAG-120: VDO Framework shall provide means to enforce the needed NIST framework requirements and raise notifications about the actions carried out.
Criticality: High Release: Intermediate-M24	FAG-130: VDO Framework shall support with tools a process for identifying vulnerabilities and the needed actions to mitigate them in a repository and crossing those indications with the configuration of the IPC device.
Criticality: High Release: Intermediate-M24	ABB-010: By using VeriDevOps technologies, requirements for new or enhanced functionality of the standard platform have to be written in such a way that they are maintainable for future extensions but also fit test cases for software module testing and overall regression test.
Criticality: High Release: Intermediate-M24	ABB-020: A VeriDevOps tool chain for requirement specification analysis should support the writing of the requirement but also make them fit for automated testing and selection of test cases/regression tests. This should fit both simulation and on-site commissioning to optimize the efficiency with the test opportunities at hand.

Criticality: High Release: Intermediate-M24	ABB-030: Requirement models and simulation models produced in VeriDevOps will offer a fast track to efficient, safe, and secure crane operations for both remote and cabin-based crane operation in an authentic environment - as close to real crane operation and realistic conditions as possible.
Criticality: High Release: Intermediate-M24	ABB-040: A system, preferably a standard, and tool to categorize the customer requirements and link them to internal requirements and systems, could aid a systematic work to determine which requirements should be handled where and to what extent they are fulfilled by the standard or project specific solutions.
Criticality: High Release: Intermediate-M24	ABB-050: Select what to evolve and innovate from assessment of the development process from requirement engineering to regression testing and deployment.
Criticality: High Release: Intermediate-M24	ABB-060: Improve the current way of writing requirements (both safety, security and the ones used in the regular process).
Criticality: High Release: Intermediate-M24	ABB-110: Increase the efficiency in understanding the requirements to be able to implement and test efficiently.
Criticality: High Release: Intermediate-M24	ABB-120: VeriDevOps methods and tools allow for designing a system on several hierarchical layers (e.g. functional and module levels). Artefacts defined on a higher level are to be linked to or be reused on lower levels. Different kinds of views on the system can be designed by using appropriate requirement types.
Criticality: High Release: Intermediate-M24	ABB-130: Validation of design against customer requirements (ensure that the design meets the customer expectations) either for safety or security requirements.

Table 1 Related Case Study Requirements

1.1.2 Services and Interfaces

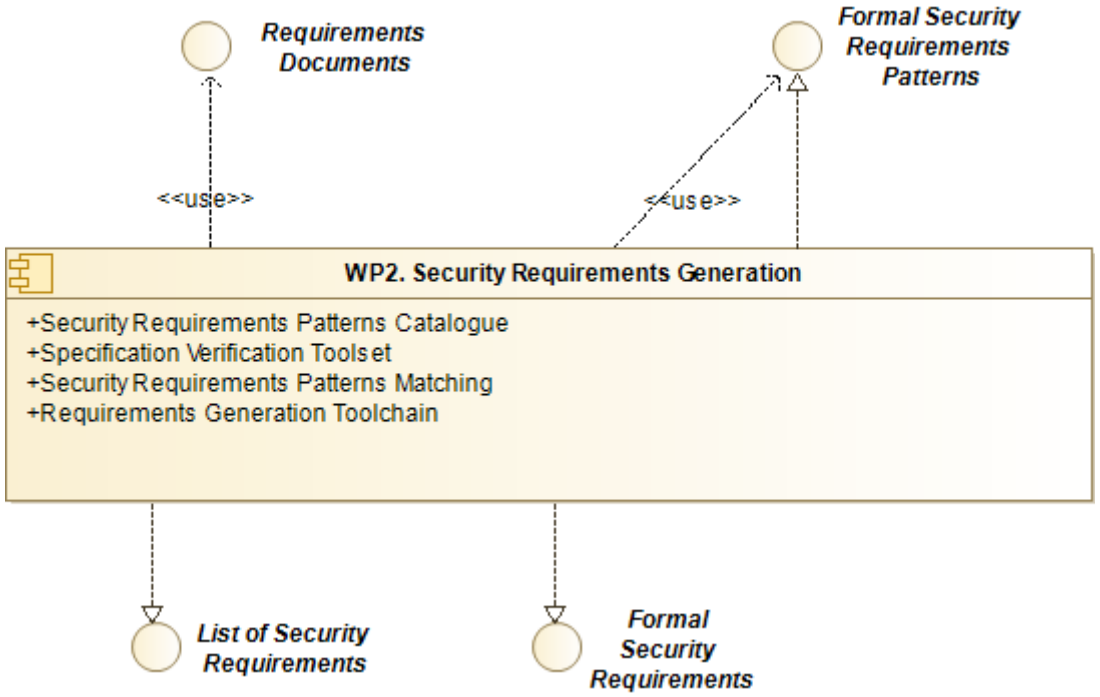

Figure 2 Services and Interfaces

1.1.2.1 Details on realized interfaces

Name	Type
List of Security Requirements	Framework Interfaces and Services
Formal Security Requirements	Framework Interfaces and Services
Formal Security Requirements Patterns	Framework Interfaces and Services

Table 2 Realized Interfaces

1.1.2.2 Details on used interfaces

Name	Type
Requirements Documents	Framework Interfaces and Services
Formal Security Requirements Patterns	Framework Interfaces and Services

Table 3 Used Interfaces

1.1.3 Subordinates

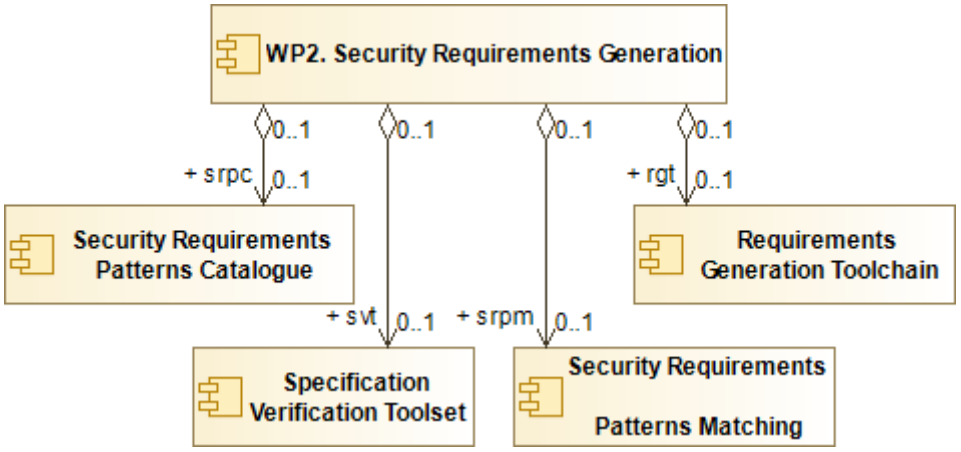

Figure 3 Subordinates

WP2 is decomposed into several subordinates as depicted in the diagram.

1.1.3.1 Details on subordinate components

Name	Description
Security Requirements Patterns Catalogue	A catalogue of patterns for formal specification of security properties.
Specification Verification Toolset	Tool for automated reasoning about security requirements and design models, in order to detect possible omissions, errors, inconsistencies and conflicts in the requirements and in the specifications. It applies both exhaustive and statistical model-checking using the formalisms and tools.
Security Requirements Patterns Matching	The pattern matching tool chain that maps the security requirements to patterns of security formal properties.
Requirements Generation Toolchain	This tool chain combines the tools for extraction, labeling, training, and classification of security requirements.

Table 4 Subordinates

1.2 WP3. Reactive Protection at Operations

As the proposal stated, the reactive protection during operation in VeriDevOps provides the main functions of continuous risk assessment including assets identification, vulnerabilities/threats detection, evaluating the risk (impact x likelihood), and providing recommendations for countermeasures to be deployed at runtime.

These reactive security mechanisms are linked to the formal vulnerability catalogue identified by a vulnerability scanning tool. This allows listing the potential vulnerabilities at different physical, application and network layers that can be exploited by hackers/intruders.

Several monitoring agents deployed in the running applications capture different traces -at

different layers- in order to extract security relevant indicators.
A rule-based detection solution will be built on top of the formal model specified in VeriDevOps and an anomaly detection solution will complement to it detect/predict any deviation for learned patterns. This last feature will rely on several ML/AI techniques including genetic algorithms.
Root Cause Analysis (RCA) is also part of the reactive security mechanisms and is a systematic process for identifying "root causes" of problems or events and an approach for responding to them. RCA is based on the idea that effective management requires more than merely "putting out fires" when problems are detected, but also finding ways to prevent them. The RCA in VeriDevOps will rely on a probabilistic decision tree approach to identify the most probable cause of any detected anomaly. Several security controls like data protection (encryption, anonymization etc.), communication securing and resiliency solutions (e.g., diversification and reflection) will be also studied in the project as potential security features to be deployed to ensure end-to-end security at runtime.

This description can be further translated in the architectural elements and linked to the case study requirements as follows.

1.2.1 Related Case Study Requirements for WP3. Reactive Protection at Operations

Properties	Definition
Criticality: High Release: Intermediate-M24	FAG-010: During software development, VeriDevOps outcome will help in testing patches before deploying them to the entire network, then reducing time and effort.
Criticality: High Release: Intermediate-M24	FAG-030: FAGOR needs an automatic procedure capable of interpreting these security requirements or recommendations such as NIST or STIG, and finding when the device is implementing them and rising alerts when devices do not fulfil them.
Criticality: High Release: Intermediate-M24	FAG-040: FAGOR needs to add to its devices a capability which can be automatic search for vulnerabilities.
Criticality: High Release: Intermediate-M24	FAG-070: FAGOR expects automated reports about anomalies on cloud environment (WP3)
Criticality: High Release: Intermediate-M24	FAG-080: FAGOR expects vulnerability scanner on edge devices (WP3)
Criticality: High Release: Intermediate-M24	FAG-100: VDO Framework shall provide means for automatic monitoring of the network and service of the MQTT server and receiving alarms or reports if anomalies are detected.

Criticality: High Release: Intermediate-M24	FAG-110: VDO Framework shall provide means to interpret the NIST framework requirements and check in the system if the requirements are implemented or not.
Criticality: High Release: Intermediate-M24	FAG-120: VDO Framework shall provide means to enforce the needed NIST framework requirements and raise notifications about the actions carried out.
Criticality: High Release: Intermediate-M24	FAG-130: VDO Framework shall support with tools a process for identifying vulnerabilities and the needed actions to mitigate them in a repository and crossing those indications with the configuration of the IPC device.
Criticality: High Release: Intermediate-M24	ABB-080: Improve the way vulnerabilities are identified, security aspects are monitored, and traces are analyzed.
Criticality: High Release: Intermediate-M24	ABB-170: Detect security breaches in the ABB system, by relying on different techniques: signature-based monitoring to identify common threats in industrial control systems, and ML/AI algorithms to detect abnormal and suspicious activities that cannot be detected using standards signatures.
Criticality: High Release: Intermediate-M24	ABB-180: Determine a list of potential countermeasures that need to be applied to enforce security at runtime.
Criticality: High Release: Intermediate-M24	ABB-190: Analyze ABB system traces at different levels (e.g. network traffic, application logs, etc.) to extract metadata, and adapt a monitoring solution to be able to parse them
Criticality: High Release: Intermediate-M24	ABB-200: Design security rules that model attacks to be avoided or properties to be achieved.
Criticality: High Release: Intermediate-M24	ABB-210: By configuring or learning the typical behaviour of a particular crane it should be possible to detect if the crane is in a hazardous state and by that avoid an accident.
Criticality: High Release: Intermediate-M24	ABB-220: By configuring or learning the typical behaviour of a particular crane it should be possible to detect a security breach or a cyber-attack.
Criticality: High Release: Intermediate-M24	ABB-230: VeriDevOps methods and devices to increase safety and security by monitoring network communication or devices' internal state.
Criticality: High Release: Intermediate-M24	ABB-240: The identification of known vulnerabilities in components and systems is needed. The known vulnerabilities refer to publicly known vulnerabilities in databases, such as, Common Vulnerability Enumeration (CVE) list published by MITRE.
Criticality: High Release: Intermediate-M24	ABB-250: Vulnerabilities identification should cover not only control software, but also software tools for development and maintenance, as well as hardware and device configuration.

Criticality: High Release: Intermediate-M24	ABB-260: VeriDevOps is to be able to monitor known vulnerabilities with the aim of ensuring security in operation.
Criticality: High Release: Intermediate-M24	FAG-020: During machine operation, VeriDevOps will produce automatic detection of known vulnerabilities and attacks, analysis of their root cause and providing effective automatic countermeasures (reaction), or manual recommendations for decision support.

Table 5 Related Case Study Requirements

1.2.2 Services and Interfaces

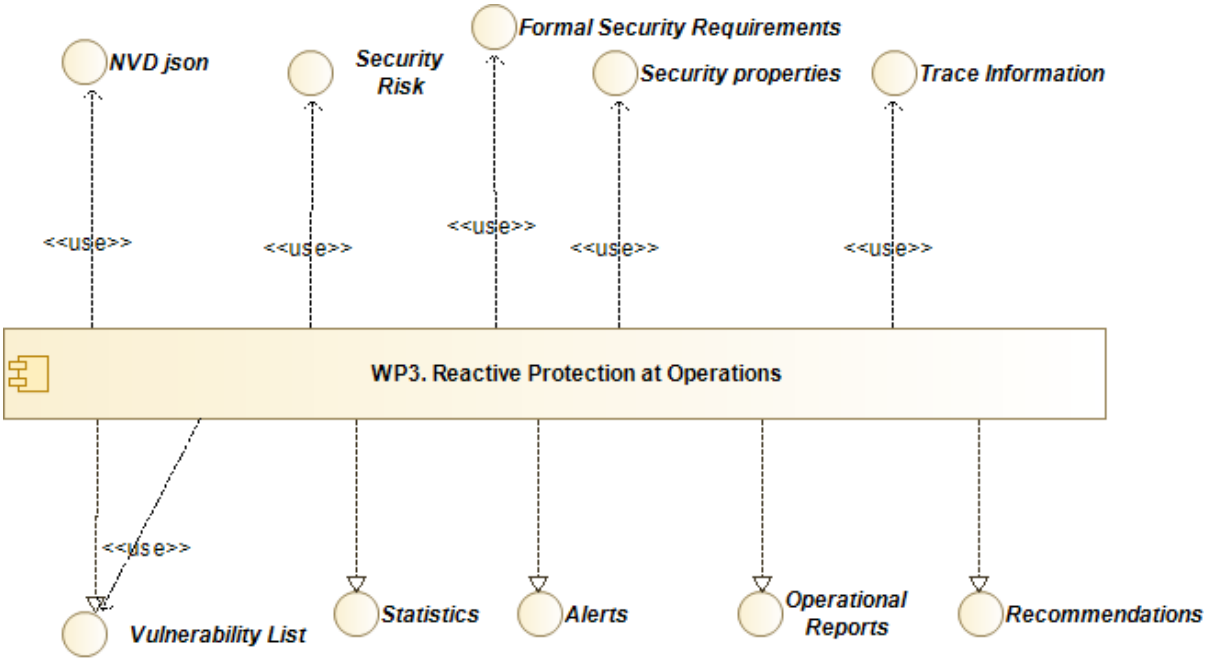

Figure 4 Services and Interfaces

The diagram illustrates the inputs and outputs of WP3 in terms of realized and used interfaces, respectively, of the VeriDevOps framework.

1.2.2.1 Details on realized interfaces

Name	Type
Statistics	Framework Interfaces and Services
Alerts	Framework Interfaces and Services
Operational Reports	Framework Interfaces and Services
Recommendations	Framework Interfaces and Services
Vulnerability List	Framework Interfaces and Services

Table 6 Realized Interfaces

1.2.2.2 Details on used interfaces

Name	Type
NVD json	Framework Interfaces and Services
Security Risk	Framework Interfaces and Services
Security properties	Framework Interfaces and Services
Trace Information	Framework Interfaces and Services
Vulnerability List	Framework Interfaces and Services
Formal Security Requirements	Framework Interfaces and Services

Table 7 Used Interfaces

1.2.3 Subordinates

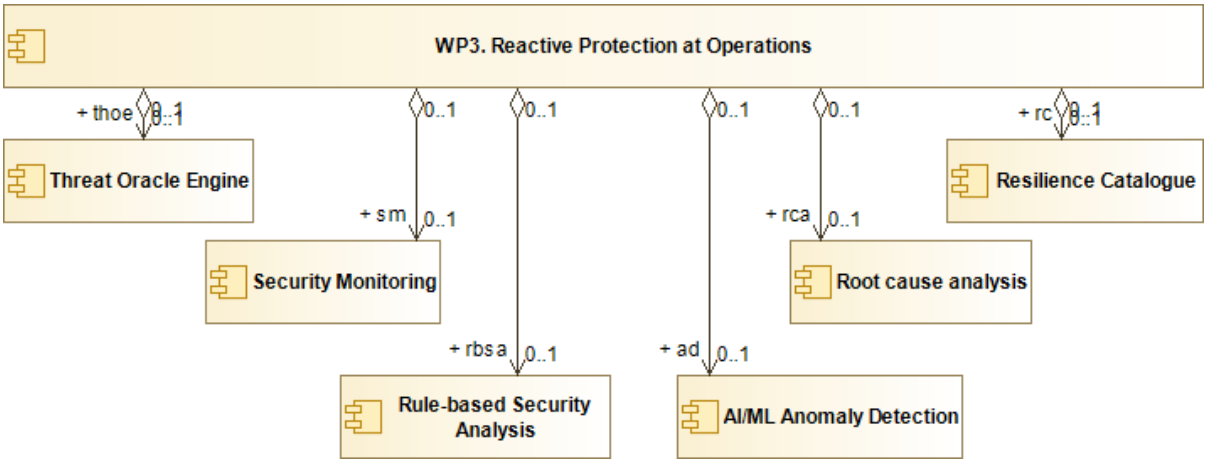

Figure 5 Subordinates

WP3 framework component will be composed of the subordinates depicted in the diagram.

1.2.3.1 Details on subordinate components

Name	Description
Threat Oracle Engine	The Threat Oracle Engine (THOE) is the tool for scanning the system for known vulnerabilities based on the information from the National Vulnerability Database (NVD) database. This THOE is a solution for industrial control system (including embedded systems with limited resources). THOE will enable: - Detection of publicly known vulnerabilities affecting software and hardware used by a product - Vulnerabilities published in different data sources, such as, NVD. Initially THOE will use NVD, but databases can be added. - Continuous monitoring for vulnerabilities and configuration - Local or remote search of vulnerabilities
Security Monitoring	The security monitoring module allows to capture different traces (network traffic, system logs etc.) in a distributed way relying on adequate security

	agents or probes. These traces will feed the rule-based security analysis and the AI/ML anomaly detection modules to detect potential security incidents and anomalies.
Rule-based Security Analysis	Security monitoring solution deployed in the operation environment to detect potential security incidents that are identified the vulnerability scanner. The detection is based on formal security properties derived from formalization of security requirements. The adaptive monitoring will allow to activate/deactivate security properties on-the-fly according to the new situation and risk. This allows to better target the security threats and optimize performance and energy consumption. The security monitoring module allows to check a set of security properties on collected traces. The analysis can be done on the fly when gathering system traces (network traffic, system logs etc.) or offline on a precaptured trace. The security properties can either specify a security rule (a requirement to be respected) or a malicious behaviour (vulnerability or attack to be avoided). In the context of VeriDevOps, the security properties will be specified in XML denoting a set of events that can be linked using temporal and logical operators inspired from LTL logic.
AI/ML Anomaly Detection	The target of the anomaly detection engine is to provide automated monitoring of the security properties against advanced threats, attacks, ransomware, viruses, breakouts and other suspicious activity that cannot be detected using standards signatures. This validation will be performed by relying on innovative probes, AI/ML based inspection enhancements to learn the normal behaviour of a system usage and detect deviations according to this baseline.
Root cause analysis	Automated Root cause analysis engine (RCA) for DevOps and QA Engineers by incorporating big data and machine learning to quickly identify security issues and their causes. This engine relies on a systematic process for identifying "root causes" of problems or events and an approach for responding to them. RCA is based on the basic idea that effective management requires more than merely "putting out fires" for problems that develop but finding a way to prevent them.
Resilience Catalogue	Basic set of security enforcement and security risk mitigation mechanisms. These mechanisms will be included in a public catalogue that can be delivered as a set of embedded libraries that can be used by security experts during development or operation following a non-intrusive approach. The whole set of enforcement mechanisms will work in an integrated way. The task will build on existing open-source solutions for example for data protection (e.g., encryption, signing, obfuscation). When possible, the tool will adopt recently proposed resilience mechanisms such as diversification and reflection. The task will research on situations in which the automatic enforcement cannot be done, and mitigation needs to come in the form of semi-automated redeployment of application components. This task will also identify and match attack threats with a potential response in case of occurrence dealing with detection and response to targets. Towards this goal, VeriDevOps will

	identify cyber incidents with impact on industrial environments, considering different malicious actors and different attack vectors, such as, semantic social engineering, network-based attacks or malware attacks. For each threat a mitigation or attack response will be defined. For instance, in case of a firmware attack, responses can be: change default network, disable DHCP or change router name and password.

Table 8 Subordinates

1.3 WP4. Prevention at Development

As the proposal stated, in VeriDevOps, we propose a set of approaches for the formal specification, analysis and verification of system requirement specifications and code. This further translates in the following architectural elements and links to the case study requirements.

1.3.1 Related Case Study Requirements for WP4. Prevention at Development

Properties	Definition
Criticality: High Release: Intermediate-M24	FAG-030: FAGOR needs an automatic procedure capable of interpreting these security requirements or recommendations such as NIST or STIG, and finding when the device is implementing them and rising alerts when devices do not fulfil them.
Criticality: High Release: Intermediate-M24	FAG-040: FAGOR needs to add to its devices a capability which can be automatic search for vulnerabilities.
Criticality: High Release: Intermediate-M24	FAG-060: FAGOR expects new testing processes or techniques on software development areas (WP4)
Criticality: High Release: Intermediate-M24	FAG-090: The defined cybersecurity tests should be compliant with international standards and certifications, such as, ISA/IEC 62443 in order to move forward a step on future certification of the IoT platform.
Criticality: High Release: Intermediate-M24	FAG-110: VDO Framework shall provide means to interpret the NIST framework requirements and check in the system if the requirements are implemented or not.
Criticality: High Release: Intermediate-M24	FAG-120: VDO Framework shall provide means to enforce the needed NIST framework requirements and raise notifications about the actions carried out.
Criticality: High Release: Intermediate-M24	FAG-130: VDO Framework shall support with tools a process for identifying vulnerabilities and the needed actions to mitigate them in a repository and crossing those indications with the configuration of the IPC device.
Criticality: High Release: Intermediate-M24	ABB-020: A VeriDevOps tool chain for requirement specification analysis should support the writing of the requirement but also make them fit for automated testing and selection of test cases/regression tests. This should

	fit both simulation and on-site commissioning to optimize the efficiency with the test opportunities at hand.
Criticality: High Release: Intermediate-M24	ABB-030: Requirement models and simulation models produced in VeriDevOps will offer a fast track to efficient, safe, and secure crane operations for both remote and cabin-based crane operation in an authentic environment - as close to real crane operation and realistic conditions as possible.
Criticality: High Release: Intermediate-M24	ABB-050: Select what to evolve and innovate from assessment of the development process from requirement engineering to regression testing and deployment.
Criticality: High Release: Intermediate-M24	ABB-070: Improve test case generation, implementation of test cases and the execution of test cases/steps by automating different steps during test design and execution.
Criticality: High Release: Intermediate-M24	ABB-090: Support nightly builds with automatic regression tests so the engineers can handle bugs and vulnerabilities faster instead of discovering these at release day when the final build is done.
Criticality: High Release: Intermediate-M24	ABB-100: Support improving the test practices related to selection and generation of effective and efficient test cases to improve test coverage (i.e., primarily functional testing and secondly module testing).
Criticality: High Release: Intermediate-M24	ABB-120: VeriDevOps methods and tools allow for designing a system on several hierarchical layers (e.g. functional and module levels). Artefacts defined on a higher level are to be linked to or be reused on lower levels. Different kinds of views on the system can be designed by using appropriate requirement types.
Criticality: High Release: Intermediate-M24	ABB-130: Validation of design against customer requirements (ensure that the design meets the customer expectations) either for safety or security requirements.
Criticality: High Release: Intermediate-M24	ABB-140: Generate test case specifications out of requirements models (improve consistency and quality of test cases and accelerate generation of test cases).
Criticality: High Release: Intermediate-M24	ABB-150: Generate test scripts out of test specifications and logic diagrams.
Criticality: High Release: Intermediate-M24	ABB-160: Prioritize and select regression test cases according to preset conditions (ensure that relevant test cases are executed at a given time).

Table 9 Related Case Study Requirements

1.3.2 Services and Interfaces

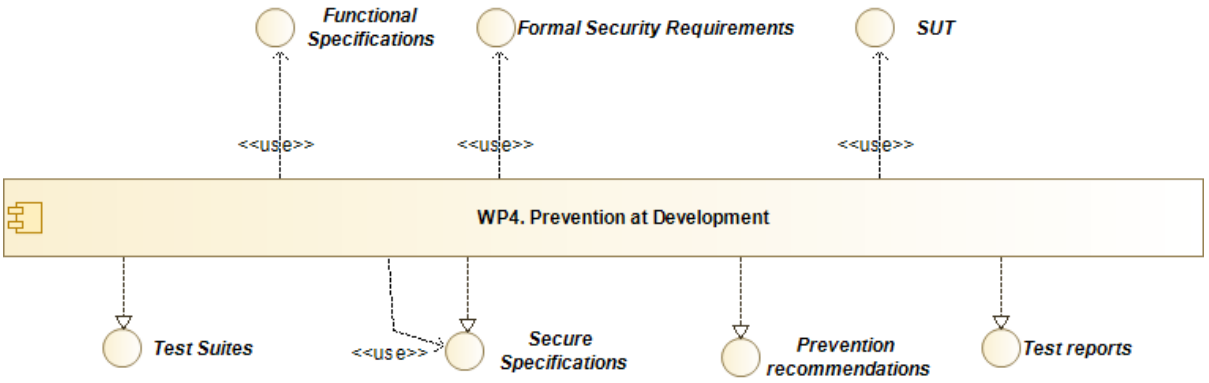

Figure 6 Services and Interfaces

The diagram illustrates the inputs and outputs of WP4 in terms of realized and used interfaces, respectively, of the VeriDevOps framework.

1.3.2.1 Details on realized interfaces

Name	Type
Test Suites	Framework Interfaces and Services
Secure Specifications	Framework Interfaces and Services
Prevention recommendations	Framework Interfaces and Services
Test reports	Framework Interfaces and Services

Table 10 Realized Interfaces

1.3.2.2 Details on used interfaces

Name	Type
Formal Security Requirements	Framework Interfaces and Services
Functional Specifications	Framework Interfaces and Services
SUT	Framework Interfaces and Services
Secure Specifications	Framework Interfaces and Services

Table 11 Used Interfaces

1.3.3 Subordinates

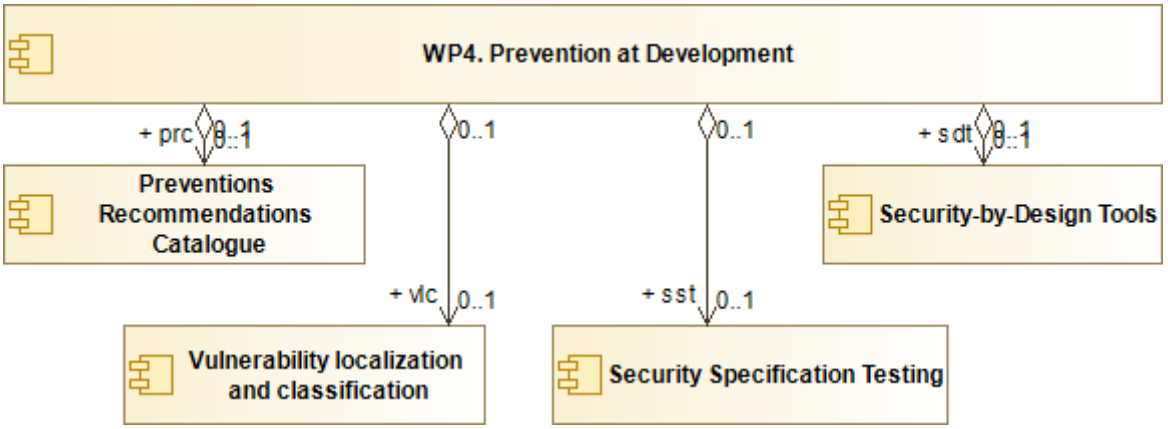

Figure 7 Subordinates

The diagram depicts the subordinate topics of WP4

1.3.3.1 Details on subordinate components

Name	Description
Preventions Recommendations Catalogue	This WP provides set of security recommendations at design at code level for identifying and fixing vulnerabilities as countermeasures. Prevention measures are based according to the type of vulnerability identified.
Vulnerability localization and classification	Tool chain for the localization of root cause of the identified vulnerabilities. The main sources of information will be the formal specifications and the execution traces of the system during the test session. We will also provide a classification of identified vulnerabilities so that we propose concrete measures for removing vulnerabilities.
Security Specification Testing	Tools for test generation from formal security specifications for both security functional testing and security vulnerability testing. For the former, we will define novel test coverage criteria for security and we will examine both online and offline test generation algorithms, using not only search-based approaches but also model-checking based test generation. For the latter, we will investigate methods like mutation testing and fuzz testing to generate tests that will uncover new (zero-day) security vulnerabilities. In addition, we will investigate methods for security regression testing to ensure that newly produced versions of the system produced in the CI/CD pipeline does not (re)introduce vulnerabilities. The main focus of the change impact analysis will be at the level of the formal specification from which the tests are automatically generated. However, the approach will be accompanied by security-related techniques for test suite minimization, test case prioritization and test case selection.

Security-by-Design Tools	Tool chain for creating secure-by-design specifications. It will instrument approaches to create the system design from case study requirements, including possible model transformations from non-formal specifications to formal ones, and for verifying the design against the security properties.

Table 12 Subordinates

2 Framework Interfaces and Services

This section gives an overview of interfaces and services realized (provided) or user (consumed) by the project tools. This section specifically focuses on the shared interfaces and services, so that the tools may integrate with each other or provide complementary functionalities.

The table below presents the list of interfaces and services as well as gives references to the tools that realize (provide) or use (consume) those interfaces or services.

Name	Description	References
Alerts	Alerts are shown at real-time and denote a violation of security rule of the detection of a malicious behavior or an anomaly.	**Realized by:** WP3. Reactive Protection at Operations, Montimage Monitoring Tool
Formal Security Requirements	Formal specification of security requirements to be produced by the Security Requirements Generation toolset.	**Realized by:** WP2. Security Requirements Generation **Used by:** WP4. Prevention at Development, WP3. Reactive Protection at Operations, CompleteTest (MDH)
Formal Security Requirements Patterns	Patterns for formal specifications of security patterns.	**Realized by:** WP2. Security Requirements Generation **Used by:** WP2. Security Requirements Generation
Functional Specifications	Specification of system functionalities in natural language or as a formal model.	**Used by:** WP4. Prevention at Development, InFuz (ABO)
IF-C++-CODE	C++ source code	**Realized by:** Modelio (SOFT)
IF-CODESYS	Codesys is a development environment for programming controller applications according to the international industrial standard IEC 61131-3. The main product of	**Used by:** SEAFOX (MDH), CompleteTest (MDH)

	the software suite is the CODESYS Development System, an IEC 61131-3 tool.	
IF-EMF-XMI	XML Metadata Interchange format implementation by Eclipse community.	**Realized by:** Modelio (SOFT) **Used by:** Modelio (SOFT)
IF-Eclipse	Eclipse provides the Rich Client Platform (RCP) for developing general purpose applications.	**Realized by:** Modelio (SOFT)
IF-JAVA-CODE	Java source code.	**Realized by:** Modelio (SOFT)
IF-JSON	In computing, JavaScript Object Notation or JSON is an open-standard file format that uses human-readable text to transmit data objects consisting of attribute–value pairs and array data types (or any other serializable value). It is a very common data format used for asynchronous browser–server communication, including as a replacement for XML in some AJAX-style systems.	
IF-LINUX-IO	Linux operating system offers different solutions for I/O. They are based on Native POSIX Threads Library (NPTL). Version: 2.19	
IF-MODEL-EDIT-GUI-RICH-CLIENT	Rich Client with Graphical User Interface for modelling usually including diagramming capabilities.	**Realized by:** Modelio (SOFT)
IF-MODEL-EDIT-GUI-WEB-CLIENT	Web Interface for modelling.	
IF-MODEL-TEXT-EDIT-GUI	Graphical User Interface for model editing in a text mode.	
IF-OMG-XMI	XML Metadata Interchange format implementation by the Object Management Group.	**Realized by:** Modelio (SOFT) **Used by:** Modelio (SOFT)
IF-PLCOpenXML	PLC code description in Open XML format.	**Used by:** SEAFOX (MDH), CompleteTest (MDH)
IF-REST	Wiki: Representational state transfer (REST) or RESTful web services is a way of providing interoperability between computer systems on the Internet.	
IF-SVN	Wiki: Apache Subversion (often abbreviated SVN, after its command name svn) is a software versioning and revision	

	control system distributed as open source under the Apache License.	
IF-WORD-DOCUMENT	Microsoft Word documents.	**Realized by:** Modelio (SOFT)
List of Security Requirements	List of security requirements statements classified according to the standard categories and if possible mapped to a set of formal requirements patterns.	**Realized by:** WP2. Security Requirements Generation
NVD json	THOE is able to scan NVD database for vulnerabilities and download them in JSON format.	**Used by:** WP3. Reactive Protection at Operations, THOE (IKER)
Operational Reports	The reports based on monitoring information at operations.	**Realized by:** WP3. Reactive Protection at Operations
Prevention recommendations	A set of prevention recommendations will be provided.	**Realized by:** WP4. Prevention at Development
Recommendations	Based on the RCA, a recommendation of countermeasures to react and mitigate a security risk. Its application can be automatic, semi-automatic or manual.	**Realized by:** WP3. Reactive Protection at Operations, Montimage Monitoring Tool
Requirements Documents	Textual documents containing security requirements expressed in natural language in English.	**Used by:** WP2. Security Requirements Generation, ARQAN (SOFT), InFuz (ABO)
SUT	System under test. This is the implementation of the system to be verified.	**Used by:** WP4. Prevention at Development, InSpeX (ABO), InFuz (ABO), CompleteTest (MDH)
Secure Specifications	This WP will provide technologies for creating secure-by-design specifications and for generating tests from them.	**Realized by:** WP4. Prevention at Development, ReSA (MDH) **Used by:** WP4. Prevention at Development, InFuz (ABO), SEAFOX (MDH)
Security Risk	The description of a security risk for example access control vulnerability, confidentiality breach or data integrity vulnerability.	**Used by:** WP3. Reactive Protection at Operations
Security properties	The security properties in MMT can define a security rules (security requirement) to be respected by the monitored system or a malicious behaviour (vulnerability or attack) to be avoided. In the first case, an alert is raised if the security property is	**Used by:** WP3. Reactive Protection at Operations, Montimage Monitoring Tool

	violated and in the second case, a alert is raised if the malicious behaviours is detected.	
Statistics	Trace analysis report with statistics and graphs.	**Realized by:** WP3. Reactive Protection at Operations, Montimage Monitoring Tool
Test Suites	The tool will generate tests from secure specifications.	**Realized by:** WP4. Prevention at Development, InSpeX (ABO), InFuz (ABO), SEAFOX (MDH), CompleteTest (MDH)
Test reports	The tools in this WP will generate test reports that will include the generate tests, passed and failed tests, coverage achieved at either specification of code, and potential vulnerabilities.	**Realized by:** WP4. Prevention at Development, InSpeX (ABO), InFuz (ABO)
Trace Information	The MMT processes the trace information. These traces can be network packets, system log entries or others.	**Used by:** WP3. Reactive Protection at Operations, Montimage Monitoring Tool
Vulnerability List	THOE provides a list of vulnerabilities according to the NVD database.	**Realized by:** WP3. Reactive Protection at Operations, THOE (IKER) **Used by:** WP3. Reactive Protection at Operations

Table 13 Framework Interfaces and Services

3 VDO Tool Components

This section provides an overview of all the tools that are developed in the projects by partners. We deliberately limited this description by very few sub-sections. In particular we focus on the features to be developed, interfaces and services provided by the tools, any breakdown on technical sub-components as well as indications on tools deployment.

3.1 Modelio (SOFT)

Modelio (SOFT): is an open-source modelling environment supporting industry standards like UML and BPMN. Modelio provides a central repository for the local model, which allows various languages (UML2 profiles such as SysML and MARTE) to be combined in the same

model, enabling abstraction layers to be managed and traceability between different model elements to be established. Modelio proposes various extension modules and can be used as a platform for building new Model-Driven Engineering (MDE) features such as code generation and reverse engineering of Java and C++. The environment enables users to build UML2 Profiles, and to combine them with a rich graphical interface for dedicated diagrams, model element property editors and action command controls.

3.1.1 Purpose of Modelio (SOFT) component

The table below provides the list of the tool features as well as their criticality, planned release date and status.

Properties	Purpose
Criticality: High Release: Baseline-M6 Status: done	MODELIO-010: Modelio shall provide system modelling capabilities.
Criticality: High Release: Intermediate-M24 Status: in progress	MODELIO-030: Modelio shall support specific security requirement modelling.
Criticality: High Release: Baseline-M6 Status: done	MODELIO-070: Modelio shall manage traceability on Modelio project level.
Criticality: High Release: Baseline-M6 Status: done	MODELIO-110: Modelio shall provide code generation facilities for specific programming languages (Java, C++, and C for example).
Criticality: High Release: Baseline-M6 Status: done	MODELIO-120: Modelio shall provide test modelling
Criticality: High Release: Baseline-M6 Status: done	MODELIO-020: Modelio shall provide extendable requirement modelling capabilities.
Criticality: High Release: Baseline-M6 Status: done	MODELIO-040: Modelio shall provide checking functionalities.
Criticality: High Release: Final-M32 Status: planned	MODELIO-050: Modelio shall provide AI/ML checking functionality and dedicated report.

Criticality: High Release: Intermediate-M24 Status: in progress	MODELIO-060: Modelio shall provide advanced query functionalities for security purpose.
Criticality: High Release: Final-M32 Status: planned	MODELIO-080: Modelio shall manage code generation with logger to be able to analyze produced trace during test and modify models accordingly to feedback.
Criticality: High Release: Baseline-M6 Status: done	MODELIO-090: Modelio shall be able to model attacks on desired system
Criticality: Medium Release: Intermediate-M24 Status: in progress	MODELIO-130: Modelio shall provide modelling support for code generation of security test and analysis.
Criticality: Medium Release: Final-M32 Status: in progress	MODELIO-150: Modelio shall provide documentation generation for report production of security analysis.
Criticality: Medium Release: Final-M32 Status: in progress	MODELIO-140: Modelio shall provide specific code generator for security test and analysis.

Table 14 Modelio (SOFT) component purpose

3.1.2 Services and Interfaces

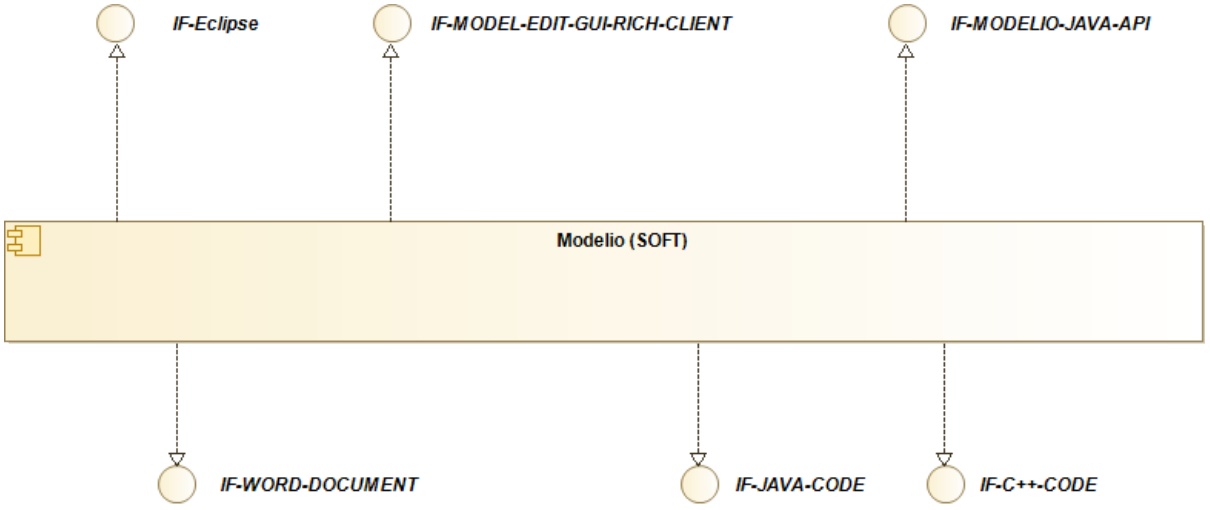

Figure 8 Services and Interfaces

3.1.2.1 Details on realized interfaces

Name	Type

IF-Eclipse	Framework Interfaces and Services
IF-MODEL-EDIT-GUI-RICH-CLIENT	Framework Interfaces and Services
IF-MODELIO-JAVA-API	Modelio (SOFT)
IF-WORD-DOCUMENT	Framework Interfaces and Services
IF-JAVA-CODE	Framework Interfaces and Services
IF-C++-CODE	Framework Interfaces and Services

Table 15 Realized Interfaces

3.1.3 Subordinates

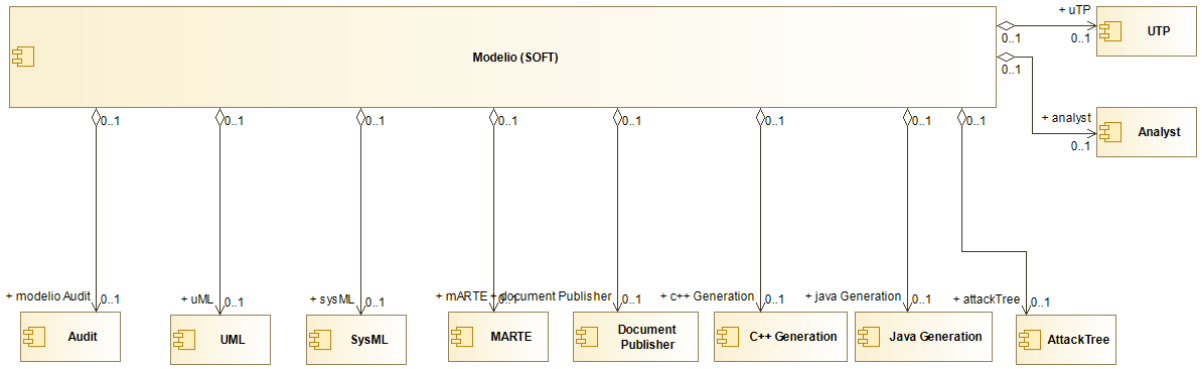

Figure 9 Subordinates

3.1.3.1 Details on subordinate components

Name	Description
Audit	Modelio provides model checking functionalities for the model editing rules.
SysML	Modelio implements SysML profile.
MARTE	Modelio implements MARTE profile.
Document Publisher	Modelio provides document generation capabilities in Word, HTML.
C++ Generation	Modelio provides C++ generation and reverse capabilities.
Java Generation	Modelio provides Java reverse and generation capabilities.
UML	Modelio implements UML metamodel.
UTP	Modelio implements UTP profile.
Analyst	Modelio provides Requirement, Risk, Goal modelling.
AttackTree	Modelio provides Attack Tree modelling.

Table 16 Subordinates

3.1.4 Relation to Framework

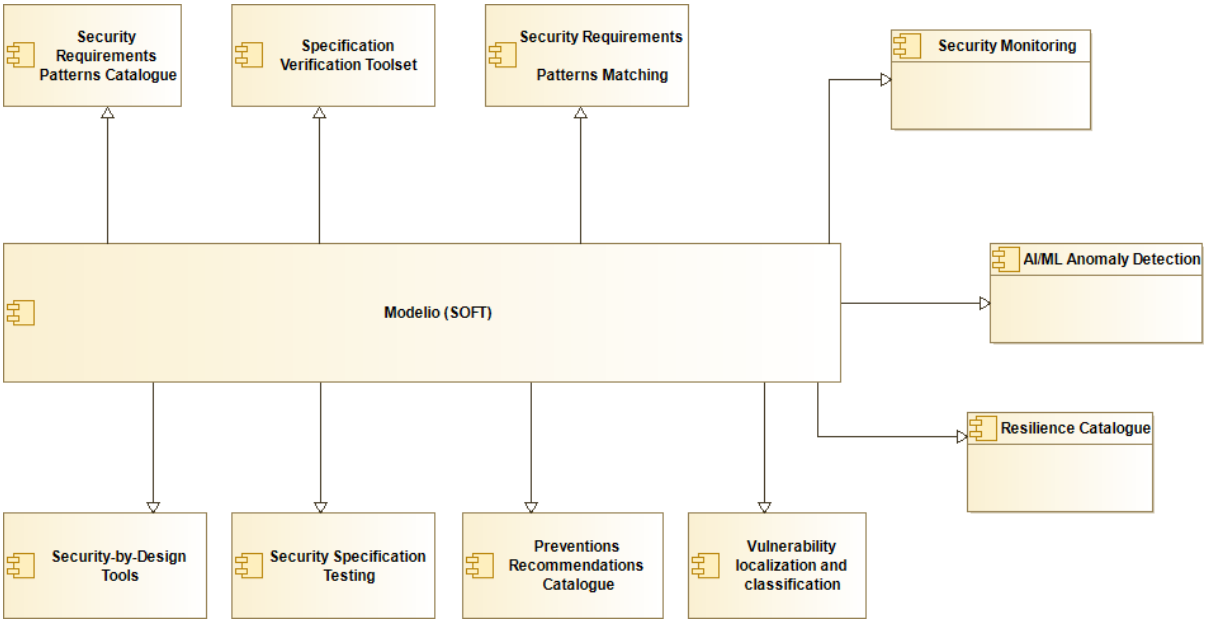

Figure 10 Relation to Framework

This diagram illustrates the relation of the tool to the features or constituent parts planned in the Framework.

3.1.4.1 Details on realized framework tools

Relation	Framework Tool	Category
Modelio should provide a catalogue of security requirements patterns.	Security Requirements Patterns Catalogue	WP2. Security Requirements Generation
Modelio should provide modelling checking facilities according to a set of security requirements.	Specification Verification Toolset	WP2. Security Requirements Generation
Modelio should be able to transform security requirements to formal properties.	Security Requirements Patterns Matching	WP2. Security Requirements Generation
Modelio should audit a design to highlight vulnerabilities.	Vulnerability localization and classification	WP4. Prevention at Development
Modelio should check design to identify vulnerabilities to propose counter measure	Preventions Recommendations Catalogue	WP4. Prevention at Development
According to a predefined set of formal security specification, Modelio should be able to generate test dedicate to a specific design.	Security Specification Testing	WP4. Prevention at Development

Modelio would generate secure system design from a set of security properties.	Security-by-Design Tools	WP4. Prevention at Development
Modelio should provide dedicated code modelling and generation for logging and FMI simulation for security purpose.	Security Monitoring	WP3. Reactive Protection at Operations
Modelio should provide AI/ML anomaly detection which could be enriched by simulations.	AI/ML Anomaly Detection	WP3. Reactive Protection at Operations
Modelio should provide a security resilience catalogue as a modelling library.	Resilience Catalogue	WP3. Reactive Protection at Operations

Table 17 Relations

3.1.5 Deployment

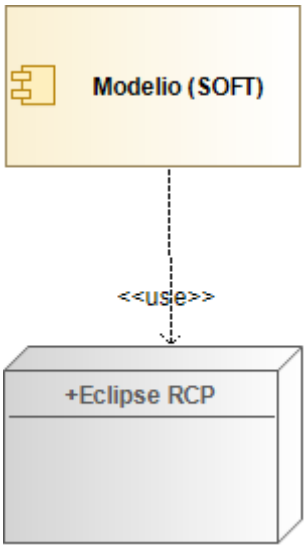

Figure 11 Deployment

3.1.5.1 Deployment details

Usage Type	Node
Modelio is implemented as an Eclipse RCP standalone tool.	Eclipse RCP

Table 18 Deployment

3.1.6 Interfaces specific to Modelio (SOFT) component

Name	Description
IF-MODELIO-JAVA-API	Modelio provides a rich Java API for extending its functionalities.

Table 19 Interfaces specific to Modelio (SOFT) component

3.2 RQCODE (SOFT)

The RQCODE is a Java-based SDK to implement formalization of requirements in the form of object-oriented notation. The requirements classes describe both the meaning of the requirement in natural language as well as provide verification and enforcement means. The object-oriented implementation provides benefits of managing requirements as code and of conducting object-oriented analysis of the requirements classes as well as automating verification and enforcement with Continuous Integration and Delivery Tools.

3.2.1 Purpose of RQCODE (SOFT) component

The table below provides the list of the tool features as well as their criticality, planned release date and status.

Properties	Purpose
Criticality: High Release: Initial-M15 Status: in progress	RQCODE-010: RQCODE shall implement the object-oriented paradigm for requirements specification.
Criticality: High Release: Initial-M15 Status: in progress	RQCODE-020: In RQCODE a requirement is represented as a class.
Criticality: High Release: Initial-M15 Status: in progress	RQCODE-030: RQCODE patterns should incapsulate textual description of requirements with enforcement and verification means.
Criticality: High Release: Intermediate-M24 Status: planned	RQCODE-040: RQCODE shall propose a ready to use catalogue of object-oriented requirements patterns.
Criticality: High Release: Intermediate-M24 Status: planned	RQCODE-050: RQCODE shall be extensible to adapt to various enforcement and verification mechanisms such as tests or windows shell scripts.

Table 20 RQCODE (SOFT) component purpose

3.2.2 Services and Interfaces

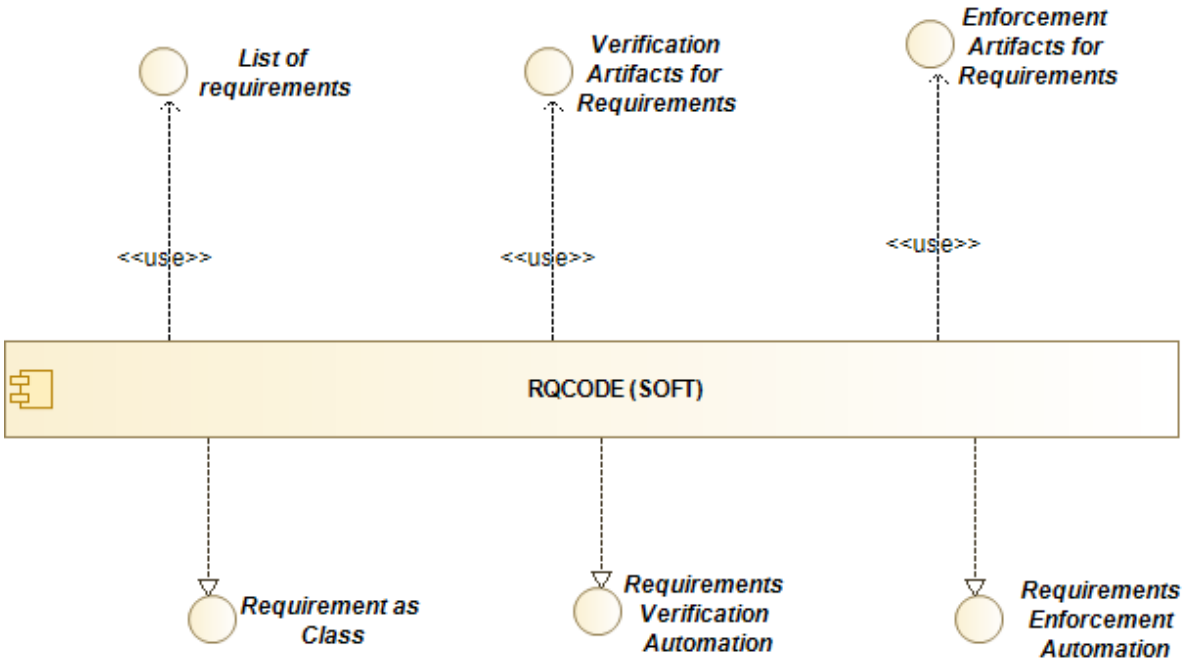

Figure 12 Services and Interfaces

3.2.2.1 Details on realized interfaces

Name	Type
Requirement as Class	RQCODE (SOFT)
Requirements Verification Automation	RQCODE (SOFT)
Requirements Enforcement Automation	RQCODE (SOFT)

Table 21 Realized Interfaces

3.2.2.2 Details on used interfaces

Name	Type
List of requirements	RQCODE (SOFT)
Verification Artifacts for Requirements	RQCODE (SOFT)
Enforcement Artifacts for Requirements	RQCODE (SOFT)

Table 22 Used Interfaces

3.2.3 Subordinates

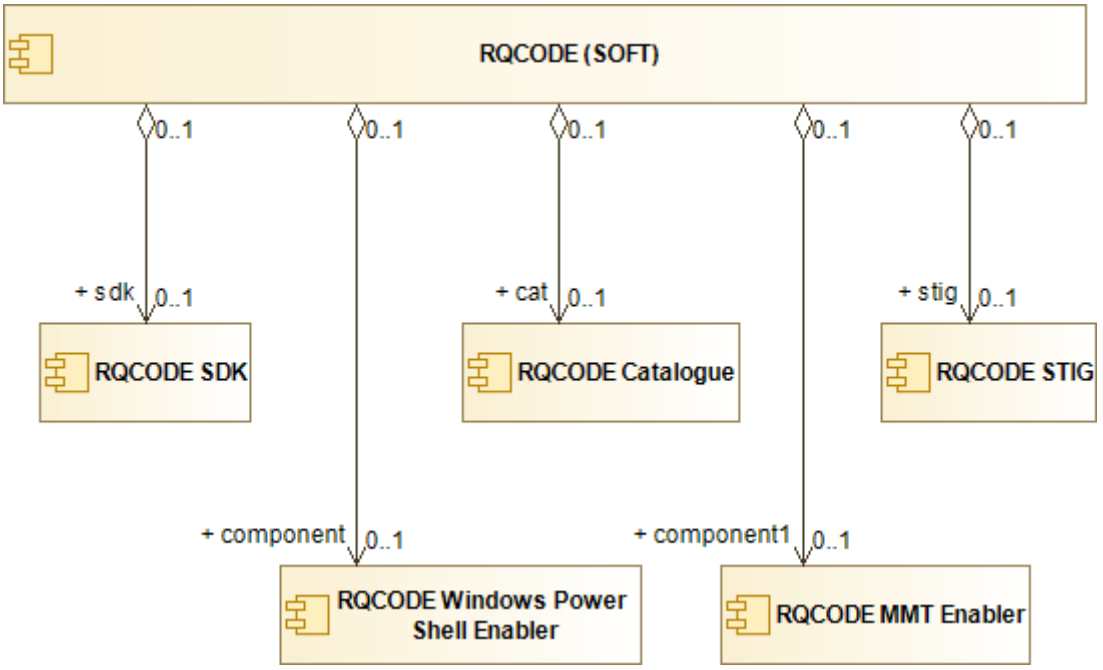

Figure 13 Subordinates

3.2.3.1 Details on subordinate components

Name	Description
RQCODE SDK	RQCODE will provide an Software Development Kit (SDK) for creating and instantiating requirements classes according to the object-oriented requirements paradigm.
RQCODE Catalogue	RQCODE will provide a repository for the object oriented requirements patterns.
RQCODE STIG	RQCODE provides specific examples for STIG rules implementation in the Requirements-as-Code paradigm.
RQCODE Windows Power Shell Enabler	RQCODE provides an enabler to run Windows Power Shell scripts for verification and enforcement.
RQCODE MMT Enabler	RQCODE has a specific enabler to configure Montimage Monitoring Tool rules as a verification means for requirements.

Table 23 Subordinates

3.2.4 Relations to the Framework

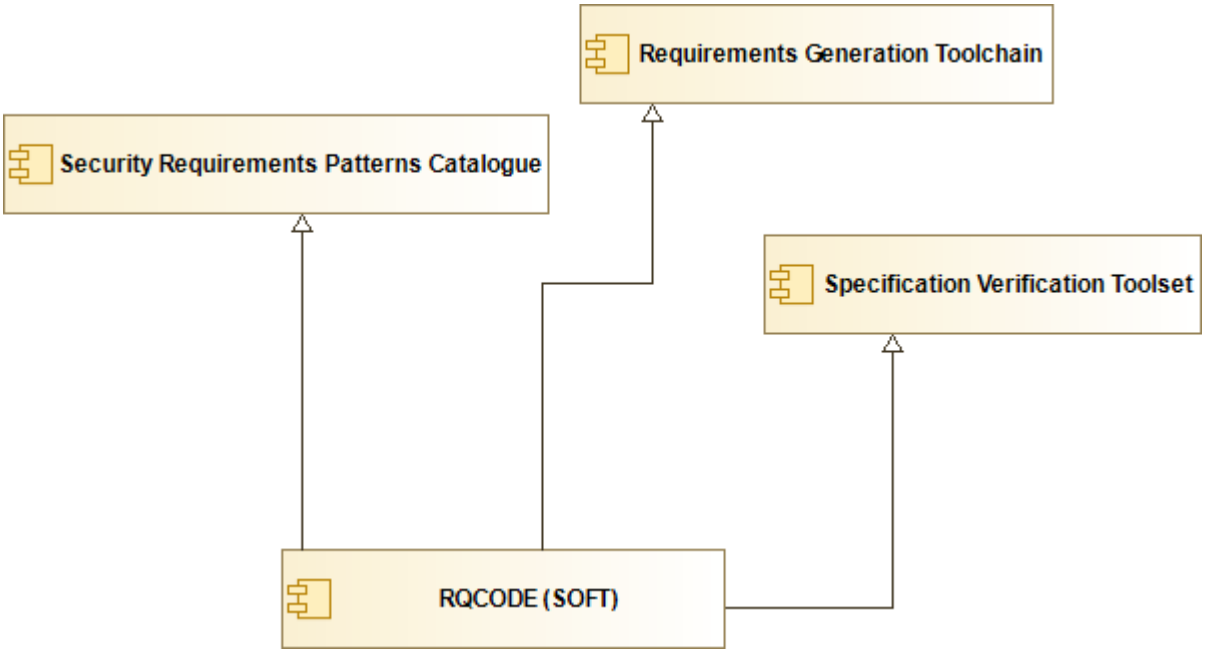

Figure 14 Relations to the Framework

3.2.4.1 Details on realized framework tools

Relation	Framework Tool	Category
RQCODE provides requirements patterns catalogue	Security Requirements Patterns Catalogue	WP2. Security Requirements Generation
RQCODE may be used to execute verification methods to verify requirements	Specification Verification Toolset	WP2. Security Requirements Generation
RQCODE is a part of the Requirements Generation Toolchain since it produces a formal representation of the requirements	Requirements Generation Toolchain	WP2. Security Requirements Generation

Table 24 Relations

3.2.5 Deployment

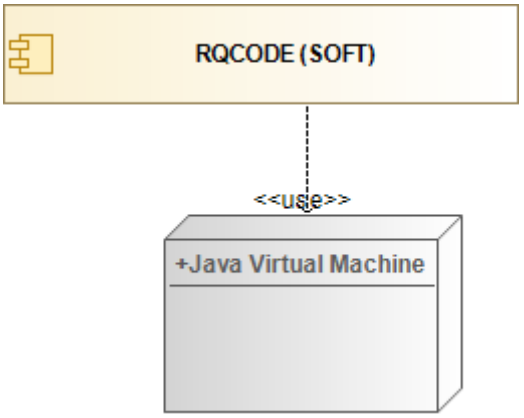

Figure 15 Deployment

3.2.5.1 Deployment details

Usage Type	Node
RQCODE SDK and the catalogue are implemented in JAVA language and require a JVM.	Java Virtual Machine

Table 25 Deployment

3.2.6 Interfaces specific to RQCODE (SOFT) component

Name	Description
List of requirements	The RQCODE need the descriptions on the requirements in natural language. These descriptions will be embedded in the requirements classes for the analysis.
Verification Artifacts for Requirements	The RQCODE needs to identify the verification specifics for each requirement. This can be directly added by the user, for example from an existing set of verification artefacts. In addition, the NLP methods can be used to deduce specific verification methods.
Enforcement Artifacts for Requirements	The RQCODE may need the specification of the enforcement methods for requirements. These artifacts can be existing ones and specified by the user. In addition, these artifacts can be deduced with automation tools from specific appropriate patterns.
Requirement as Class	RQCODE implements the Requirements as Code paradigm whereas requirements are represented as classes incapsulating the textual descriptions of the requirements along with verification and enforcement means.
Requirements Verification Automation	RQCODE will provide requirements automation capability by executing verification methods associated with requirements.

Requirements Enforcement Automation	RQCODE will provide requirements enforcement capability by executing the enforcement methods associated with requirements.

Table 26 Interfaces specific to RQCODE (SOFT) component

3.3 ARQAN (SOFT)

The ARQAN tool is dedicated to application in the extraction/classification/patterns recommendation problems. The tool supports the following process that is presented as a combination of designed sub-technologies and integrated as a single system for an end-to-end solution.

Possible NLP tasks: Classification, Extraction, Patterns recognition

The vital stage is assigned with a specific number and its assigned purpose, whereas supporting steps are subscribed only with their functionality, which means that they are not active at the stage of use.
1) The MMTend-to-end solution is started with dragging a document in a PDF format (so far only this format is supported). The system splits the document up into sentences removing some uninformative elements from the text like stop words, comas, brackets etc. Those sentences will be digested afterward directly to the first transformer model, which is DistilBERT in our case. You should notice that this architecture was pre-trained on the PURE dataset to differentiate sentences between requirements and non-requirements. This dataset was initially extracted and marked up accordingly.
2) In the next step, the extracted list of sentences goes to the T5 transformer, which is focused on classifying extracted requirements between security and non-security. For preparing this architecture PROMISE dataset was used for training purposes.
3) Afterward, we can extract similar patterns based on the extracted security requirements. Specifically, we obtain a numerical vector representation for each sentence. Those vectors can be compared with an initially preserved database applying a cosine similarity measure. Practically we compare each security requirement embedding one-by-one with the whole list of preliminary extracted embeddings (e.g. STIG). Such a method is widespread in the NLP area when we want to get a numerical measure of textual and semantic similarity. Those patterns can serve as a source of auxiliary information (e.g. about STIG requirements) about requirements like risk/security requirements description, severity, actual platforms etc.

Each presented stage might be applied separately as well as its model depending upon the task that is being solved.

3.3.1 Purpose of ARQAN (SOFT) component

The table below provides the list of the tool features as well as their criticality, planned release date and status.

Properties	Purpose
Criticality: High **Release:** **Initial-M15** **Status: done**	ARQN-010: The tool shall process in the PDF format.
Criticality: High **Release:** **Initial-M15** **Status: in progress**	ARQN-020: The tool shall extract security requirements from documents in PDF.
Criticality: Medium **Release:** **Intermediate-M24** **Status: in progress**	ARQN-030: The tool shall categorize requirements according to the standard categories of security requirements.
Criticality: Medium **Release:** **Intermediate-M24** **Status: planned**	ARQN-040: The tool shall propose relevant recommendations to prevent security vulnerabilities based on STIG guidelines.
Criticality: Medium **Release: Final-M32** **Status: planned**	ARQN-050: ARQAN shall provide User Interface for navigating lists of classified security requirements, and their mappings to recommendations and patterns.
Criticality: Medium **Release: Final-M32** **Status: planned**	ARQN-060: ARQAN shall provide User Interface for classification pipeline administration.
Criticality: Medium **Release: Final-M32** **Status: planned**	ARQN-070: ARQAN shall provide patterns matching for implementing mappings of security requirements in English with patterns of formal security requirements.

Table 27 ARQAN (SOFT) component purpose

3.3.2 Services and Interfaces

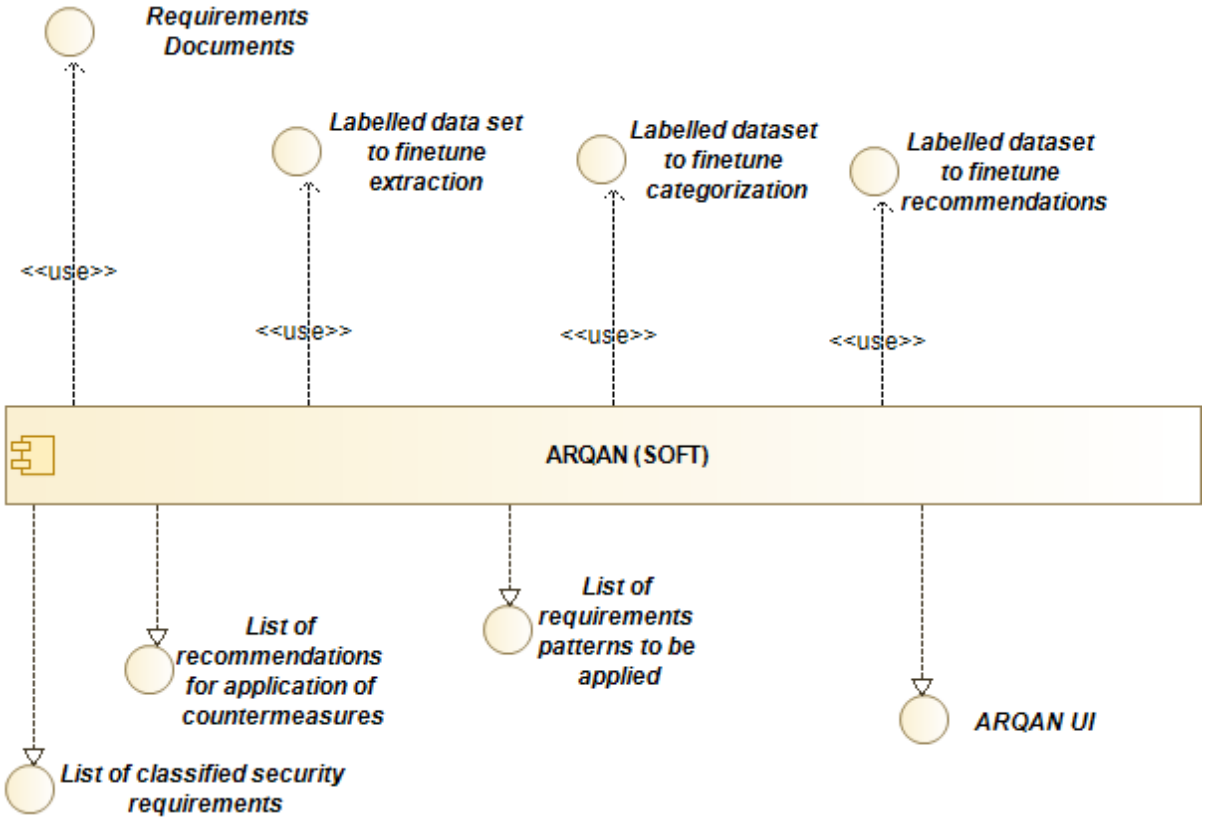

Figure 16 Services and Interfaces

3.3.2.1 Details on realized interfaces

Name	Type
List of classified security requirements	ARQAN (SOFT)
List of recommendations for application of countermeasures	ARQAN (SOFT)
ARQAN UI	ARQAN (SOFT)
List of requirements patterns to be applied	ARQAN (SOFT)

Table 28 Realized Interfaces

3.3.2.2 Details on used interfaces

Name	Type
Requirements Documents	Framework Interfaces and Services
Labelled data set to finetune extraction	ARQAN (SOFT)
Labelled dataset to finetune categorization	ARQAN (SOFT)
Labelled dataset to finetune recommendations	ARQAN (SOFT)

Table 29 Used Interfaces

3.3.3 Subordinates

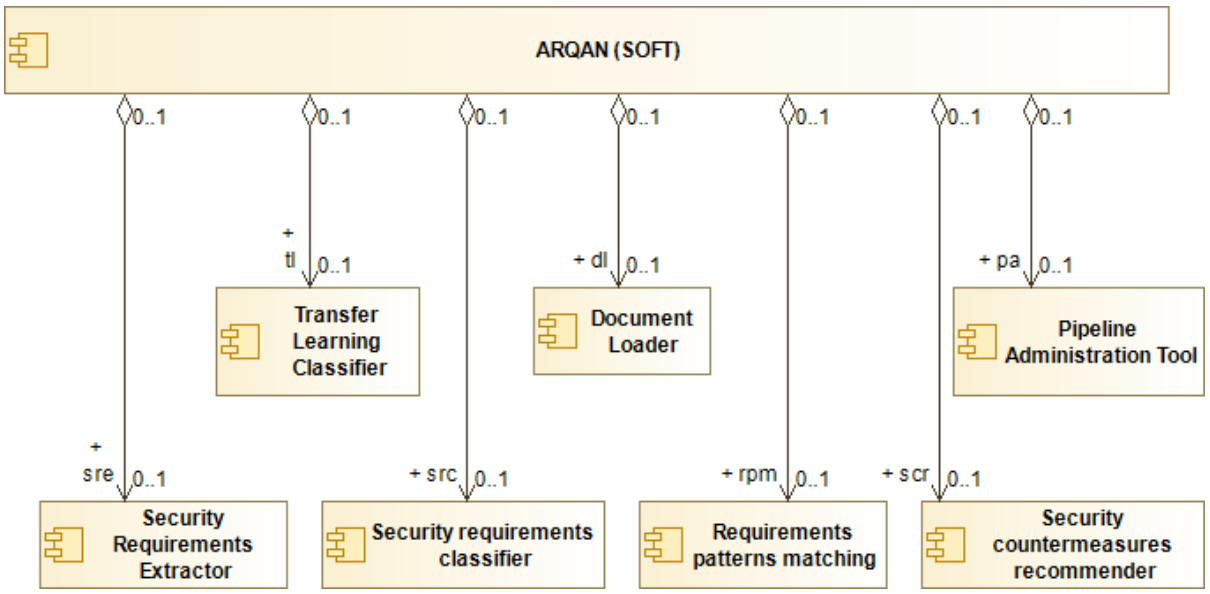

Figure 17 Subordinates

3.3.3.1 Details on subordinate components

Name	Description
Transfer Learning Classifier	The tool is based on the applying ML classifier based on transfer learning architecture, i.e. BERT, T5 or others.
Document Loader	The Document Loader pre-process an input document to extract sentences to be classified.
Pipeline Administration Tool	The pipeline administrator is used to setup various parameters of the classification tool chain such as filters to be used and thresholds to be applied as well as execution of re-training and managing various versions of models.
Security requirements classifier	This tool categorizes security in several categories that are defined at the finetuning stage.
Security Requirements Extractor	This tool classifies input sentences in English as related to security requirements or not.
Requirements patterns matching	This tool provides recommendations on mapping security requirements or categories of security requirements to the available patterns of formal security requirements.
Security countermeasures recommender	This tool provides recommendations for related countermeasures against potential vulnerabilities relevant in the context of security requirements analysis.

Table 30 Subordinates

3.3.4 Relations to the Framework

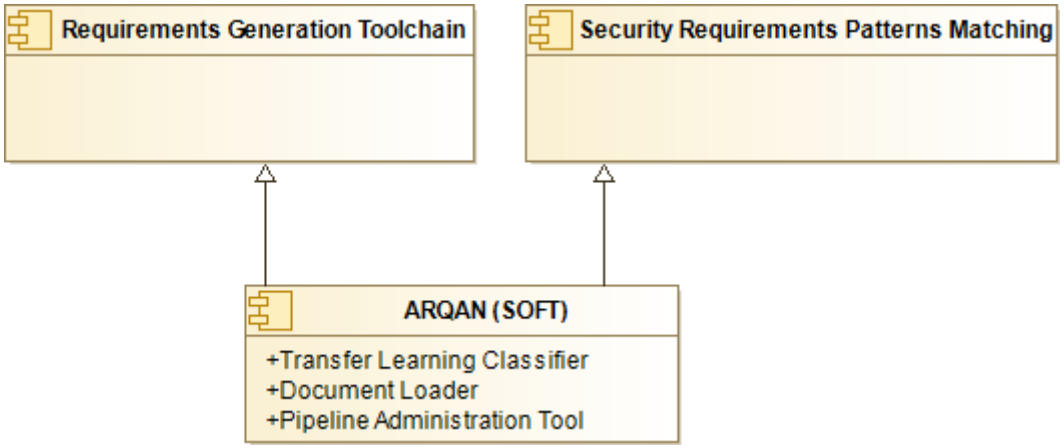

Figure 18 Relations to the Framework

3.3.4.1 Details on realized framework tools

Relation	Framework Tool	Category
ARQAN is a part of the requirements generation tool chain and serves at the initial stage where the security requirements are detected in and extracted from the documents in natural language.	Requirements Generation Toolchain	WP2. Security Requirements Generation
ARQAN includes various classifiers that may be used for pattern matching, in particular, in the countermeasure recommendation tasks.	Security Requirements Patterns Matching	WP2. Security Requirements Generation

Table 31 Relations

3.3.5 Deployment

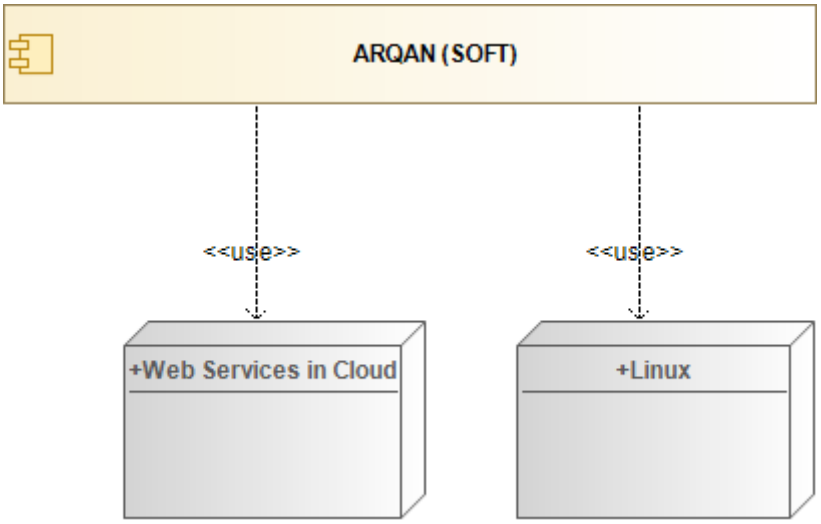

Figure 19 Deployment

3.3.5.1 Deployment details

Usage Type	Node
ARQAN may be used as a Web Service to benefit of calculation resources in the Cloud for better performance in training and executing the classification.	Web Services in Cloud
ARQAN may be deployed on Linux server of the user.	Linux

Table 32 Deployment

3.3.6 Interfaces specific to ARQAN (SOFT) component

Name	Description
Labelled data set to finetune extraction	In ARQAN the extraction of requirements may be finetuned by applying custom datasets.
Labelled dataset to finetune categorization	In ARQAN the categorization of requirements may be finetuned by applying custom datasets.
Labelled dataset to finetune recommendations	In ARQAN the recommendations based on requirements analysis may be finetuned by applying custom datasets.
List of classified security requirements	List of security requirements statements classified according to the standard categories and if possible mapped to a set of formal requirements patterns.
List of recommendations for application of countermeasures	ARQAN provides list of recommendations against vulnerabilities related to security requirements.

List of requirements patterns to be applied	ARQAN provides list of options for mapping security requirements to relevant patterns of formal requirements patterns.
ARQAN UI	ARQAN provides User Interface for administration, processing documents and obtaining the lists of classified requirements and recommendations.

Table 33 Interfaces specific to ARQAN (SOFT) component

3.4 Montimage Monitoring Tool

Montimage Monitoring Tool (MMT) is a monitoring solution that allows capturing and analyzing network traffic. It can be used to understand how network is used (protocols, applications and users) and detect potential security and performance incidents. The MMT tool has a plugin architecture that allows its easy extension in order to target new protocols and, more generally, any input data to be analyzed. In the context of VeriDevOps, MMT will be extended to analyse system and application logs provided by industrial systems, to check a set a predefined security properties to detect security incidents. It will also integrate an anomaly detection mechanism that relies on ML/AI algorithms to identify potential drifts and deviations from a learned baseline. A root cause analysis to determine the origins of such security incident and anomalies will allow to better select the countermeasure that can be applied among a set of potential resilience catalogue.

3.4.1 Purpose of Montimage Monitoring Tool component

The table below provides the list of the tool features as well as their criticality, planned release date and status.

Properties	Purpose
Criticality: High Release: Baseline-M6 Status: done	MMT-010: Solution to monitor network traffic. It allows to analyse more than 600 different protocols and to provide a deep insight of the network usage, performance and security.
Criticality: High Release: Initial-M15 Status: in progress	MMT-020: Security monitoring module allows to retrieve relevant security attributes from system and application traces in the context of VeriDevOps case studies. The security module relies on a plugin architecture to add new types of traces in the MMT monitoring solution.
Criticality: High Release: Initial-M15 Status: in progress	MMT-030: The rule-based analysis module relies on adequate security properties that are specified in XML and inspired from LTL.

Criticality: High Release: Initial-M15 Status: in progress	MMT-040: The ML/AI analysis module allow to detect anomalies of case study traces. This also target encrypted traffic analysis that is relevant for secure communications.
Criticality: Medium Release: Intermediate-M24 Status: planned	MMT-050: The root cause analysis module allows to determine the origin of security incidents.
Criticality: Medium Release: Intermediate-M24 Status: planned	MMT-060: A list of recommendations are proposed to mitigate the risk after a security incident. This list a part of a resilience catalogue to be defined during the project.
Criticality: High Release: Final-M32 Status: planned	MMT-070: Integration of all the previous modules with a friendly user interface.

Table 34 Montimage Monitoring Tool component purpose

3.4.2 Services and Interfaces

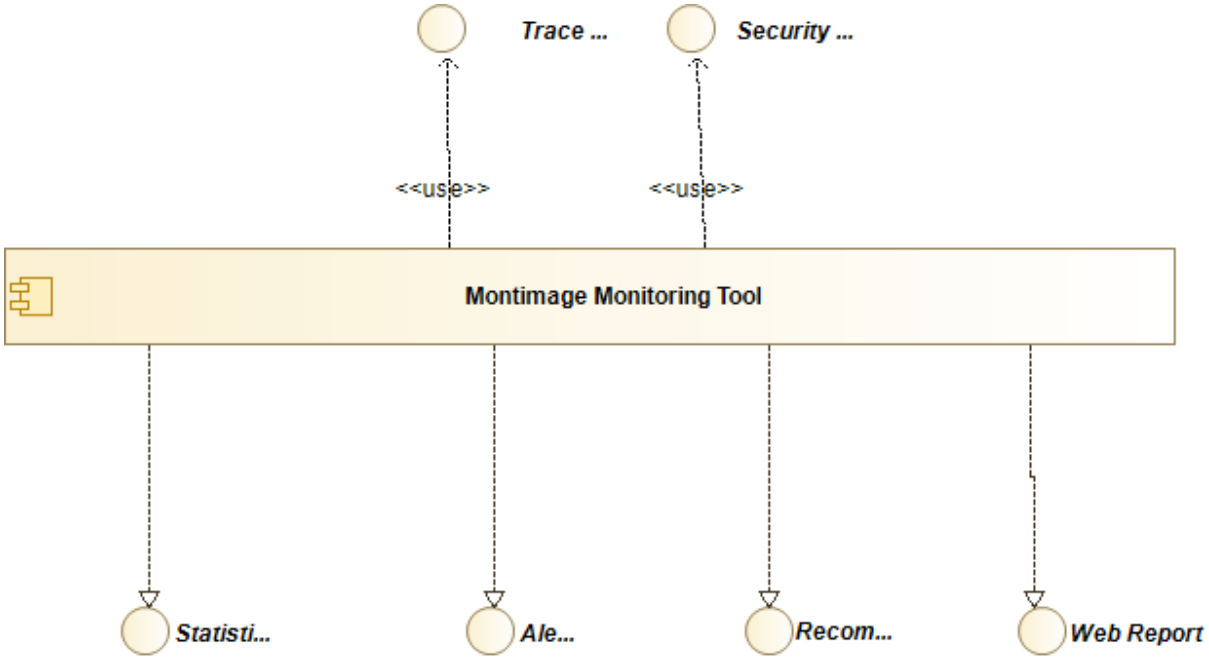

Figure 20 Services and Interfaces

3.4.2.1 *Details on realized interfaces*

Name	Type
Statistics	Framework Interfaces and Services
Alerts	Framework Interfaces and Services
Web Report	Montimage Monitoring Tool
Recommendations	Framework Interfaces and Services

Table 35 Realized Interfaces

3.4.2.2 *Details on used interfaces*

Name	Type
Trace Information	Framework Interfaces and Services
Security properties	Framework Interfaces and Services

Table 36 Used Interfaces

3.4.3 Subordinates

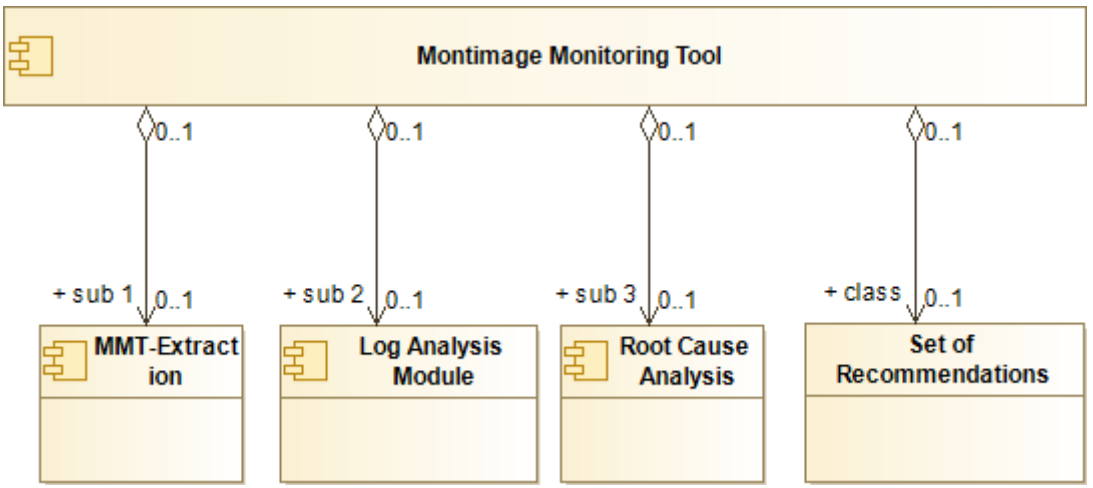

Figure 21 Subordinates

3.4.3.1 *Details on subordinate components*

Name	Description
MMT-Extraction	MMT-Extraction is the module of MMT that allows to extract security relevant attributes from analysed traces. MMT-Extraction has a plugin architecture that allow to extend it and add new type of input data coming from different security agents and system interfaces.
Log Analysis Module	Log analysis allows to check the security properties and to detect potential security incident and anomalies based on ML/AI algorithms.

Root Cause Analysis	Root Cause Analysis module intends to determine the origins of security incidents.
Set of Recommendations	Recommendations of reaction (in order to mitigate security risk) is proposed by the tool among a set of potential recommendations in a resilience catalogue.

Table 37 Subordinates

3.4.4 Relation to framework

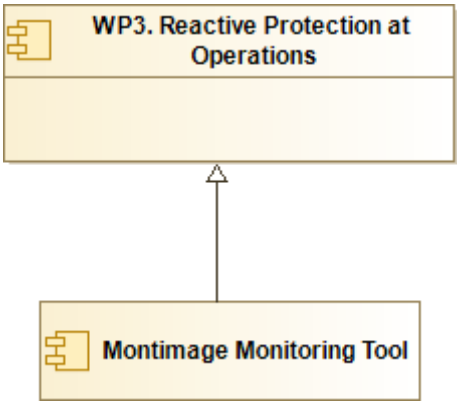

Figure 22 Relation to framework

3.4.4.1 Details on realized framework tools

Relation	Framework Tool	Category
MMT implements the features that has been described in the VeriDevOps framework under the Reactive Protection at Operations toolset.	WP3. Reactive Protection at Operations	Step3.VeriDevOps Framework

Table 38 Relation to Framework

3.4.5 Deployment

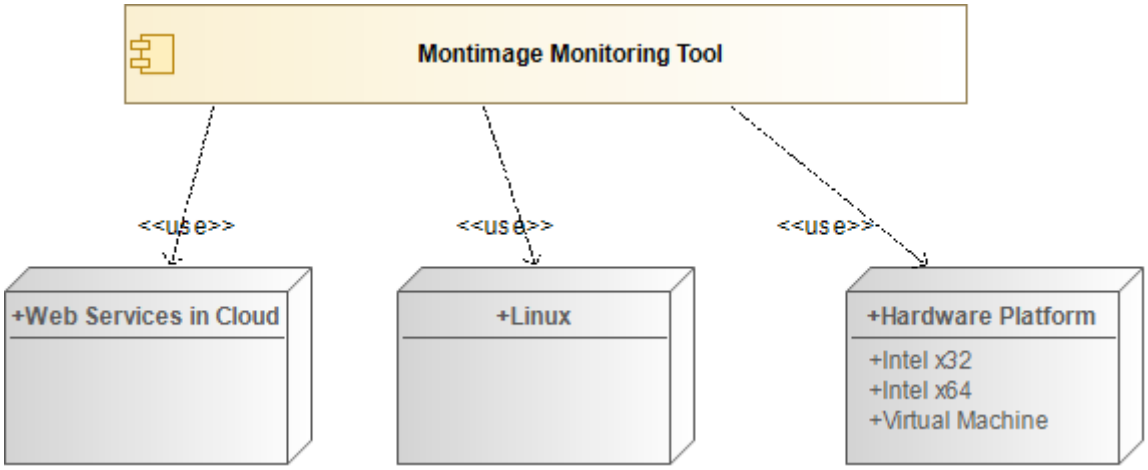

Figure 23 Deployment

3.4.5.1 Deployment details

Usage Type	Node
MMT deploys to Linux PCs.	Linux
Users may use the Cloud version of MMT.	Web Services in Cloud
MMT can be provided as hardware device.	Hardware Platform

Table 39 Deployment

3.4.6 Interfaces specific to Montimage Monitoring Tool component

Name	Description
Web Report	MMT provide Web Interface for all the information including the runtime monitoring and historical data. The Web Interface can be used to configure the tool including the rules, reports, alerts.

Table 40 Interfaces specific to Montimage Monitoring Tool component

3.5 THOE (IKER)

The Threat Oracle Engine (THOE) is the tool for scanning the system for known vulnerabilities based on the information from the NVD database.

This THOE is a solution for industrial control system (including embedded systems with limited resources). THOE will enable:

- Detection of publicly known vulnerabilities affecting software and hardware used by a product
- Vulnerabilities published in different data sources, such as, NVD. Initially THOE will use NVD, but databases can be added.

- Continuous monitoring for vulnerabilities and configuration
- Local or remote search of vulnerabilities.

3.5.1 Purpose of THOE (IKER) component

The table below provides the list of the tool features as well as their criticality, planned release date and status.

Properties	Purpose
Criticality: High Release: Baseline-M6 Status: done	THOE-010: THOE shall use NVD (National Vulnerability Database) as the main source for searching publicly-known vulnerabilities. https://nvd.nist.gov/
Criticality: High Release: Initial-M15 Status: in progress	THOE-020: THOE shall search for vulnerabilities in NVD by means of hardware and software Common Platform Enumeration identifiers, that is, it shall provide query functionalities given a certain CPE identifier
Criticality: High Release: Initial-M15 Status: in progress	THOE-030: THOE shall enable the visualization of vulnerabilities
Criticality: High Release: Intermediate-M24 Status: planned	THOE-040: THOE shall generate a report with the list of vulnerabilities and corresponding CVSS values that measures the vulnerability severity
Criticality: High Release: Intermediate-M24 Status: planned	THOE-050: THOE shall support the import of CPE-based asset inventory
Criticality: Medium Release: Final-M32 Status: planned	THOE-060: THOE shall support the visualization and download of the logs information
Criticality: Low Release: Final-M32 Status: planned	THOE-070: THOE shall enable the configuration of a scheduler for searching vulnerabilities for a given product

Table 41 THOE (IKER) component purpose

3.5.2 Functional Interfaces (Integration Means)

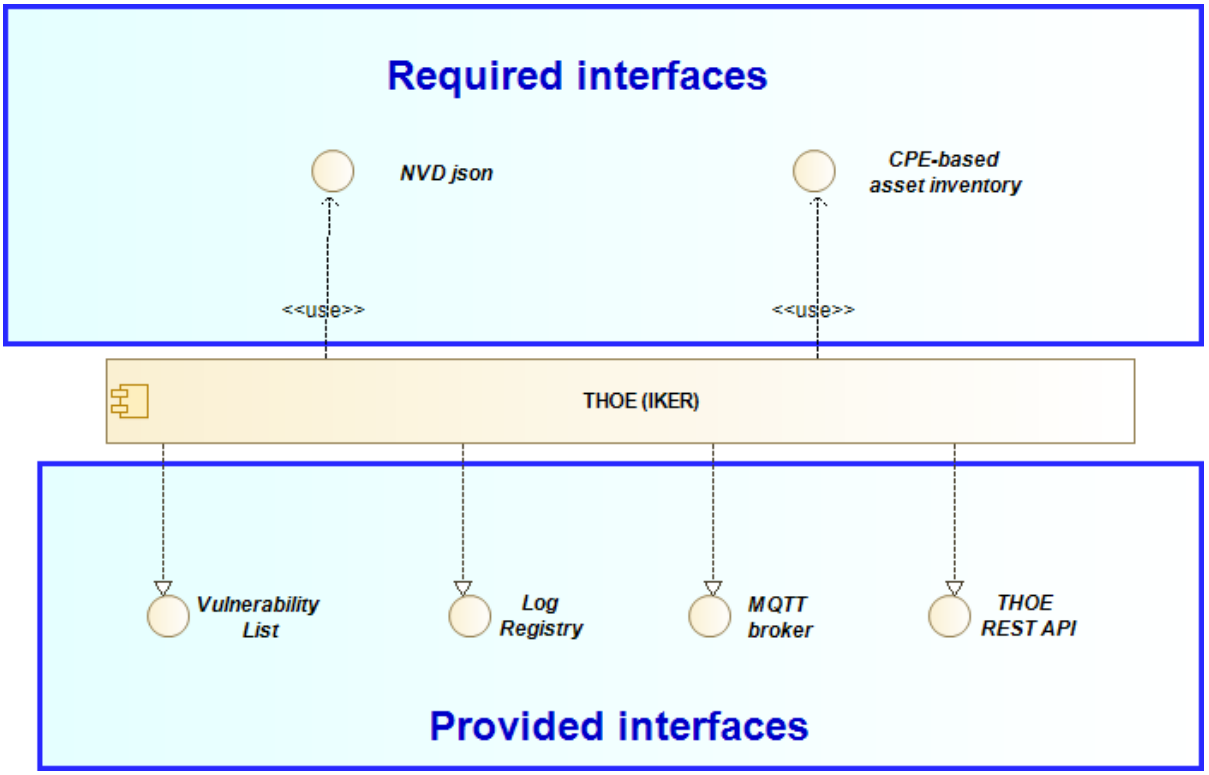

Figure 24 Functional Interfaces (Integration Means)

Provided services

3.5.2.1 Details on realized interfaces

Name	Type
Vulnerability List	Framework Interfaces and Services
Log Registry	THOE (IKER)
MQTT broker	THOE (IKER)
THOE REST API	THOE (IKER)

Table 42 Realized Interfaces

3.5.2.2 Details on used interfaces

Name	Type
NVD json	Framework Interfaces and Services
CPE-based asset inventory	THOE (IKER)

Table 43 Used Interfaces

3.5.3 Subordinates

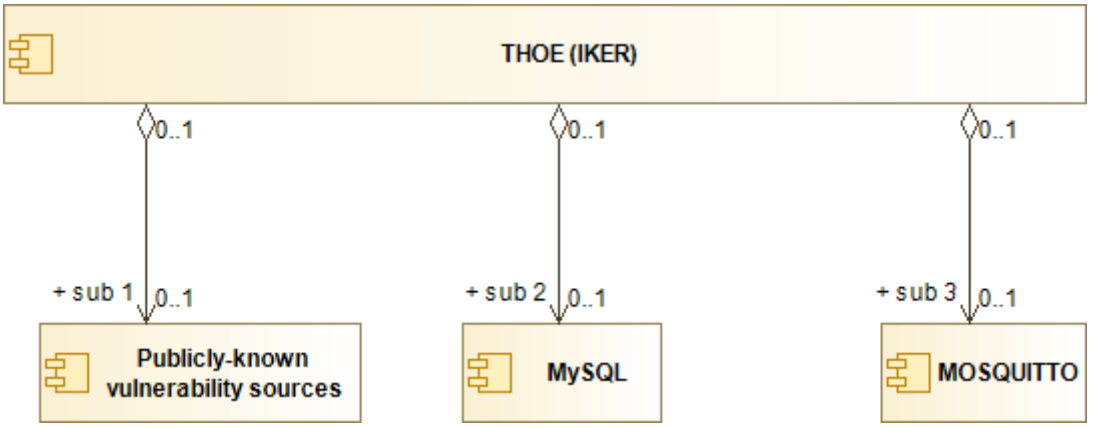

Figure 25 Subordinates

3.5.3.1 Details on subordinate components

Name	Description
Publicly-known vulnerability sources	This sub-component searches inside of publicly known databases or sources for new vulnerabilities. NVD will be used as the main vulnerability source but other sources can be added.
MySQL	Three MySQL databases will be used for: - database for threat finder configuration - service database - management database
MOSQUITTO	Mosquitto message broker that implements the MQTT protocol will be used. Mosquitto is lightweight and is suitable for use on all devices with limited and non-limited resources.

Table 44 Subordinates

3.5.4 Relations to the Framework

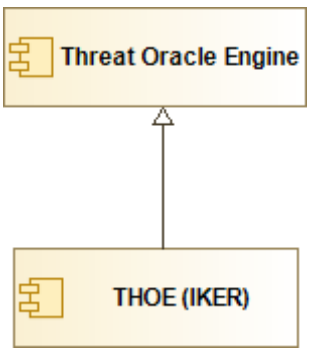

Figure 26 Relations to the Framework

3.5.4.1 Details on realized framework tools

Relation	Framework Tool	Category
The THOE by IKERLAN provides features that were planned in the VeriDevOps framework and described in the proposal under the name of a generic threat oracle.	Threat Oracle Engine	WP3. Reactive Protection at Operations

Table 45 Relations to Framework

3.5.5 Deployment diagram

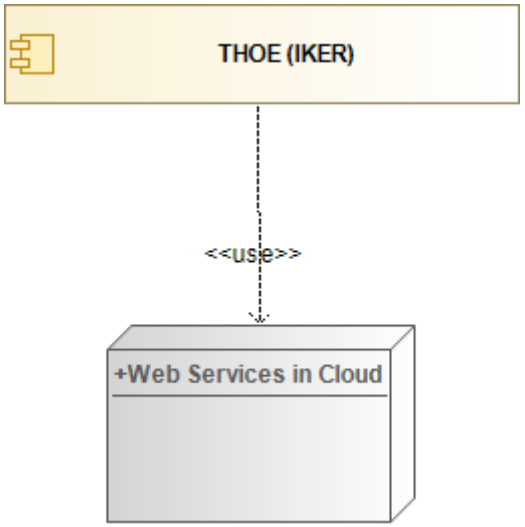

Figure 27 Deployment diagram

3.5.5.1 Deployment details

Usage Type	Node
THOE is accessed as a Web Service.	Web Services in Cloud

Table 46 Deployment

3.5.6 Interfaces specific to THOE (IKER) component

Name	Description
CPE-based asset inventory	THOE will be based on defined Common Platform Enumeration (CPE) identifiers for each hardware and software. THOE will use a CPE-based asset inventory for hardware and software to ensure that not only the operating system and installed applications are considered for vulnerability detection, but also libraries, packages, and so on. This asset inventory shall be updated whenever a software update/upgrade is performed.

Log Registry	The THOE provides a log registry. These logs are all traces generated in THOE for a specific system
MQTT broker	A MQTT broker runs in THOE so that any equipment can be connected to for getting vulnerability list and log information. Authentication/authorization will be required
THOE REST API	A Representational State Transfer (REST) API is an interface that support sets of HTTP operations (methods), which provide a way to interact with THOE. By means of this interface, VeriDevOps tools or endpoints/devices can access to THOE for checking current vulnerabilities, adding a new software asset to be scanned, modify vulnerability scheduler, and so on.

Table 47 Interfaces specific to THOE (IKER) component

3.6 InSpeX (ABO)

A tool that generates tests efficiently for systems with large input spaces. It tries to optimize the test generation by focusing on the input regions perceived to be more error prone.

3.6.1 Purpose of InSpeX (ABO) component

The table below provides the list of the tool features as well as their criticality, planned release date and status.

Properties	Purpose
Criticality: High Release: Final-M32 Status: in progress	ABO-InSpEx-010: shall provide automated and targeted functional, safety and security test generation
Criticality: Medium Release: Final-M32 Status: planned	ABO-InSpEx-020: shall provide human friendly test reports
Criticality: Medium Release: Final-M32 Status: planned	ABO-InSpEx-030: shall provide machine-readable test scripts

Table 48 InSpeX (ABO) component purpose

3.6.2 Services and Interfaces

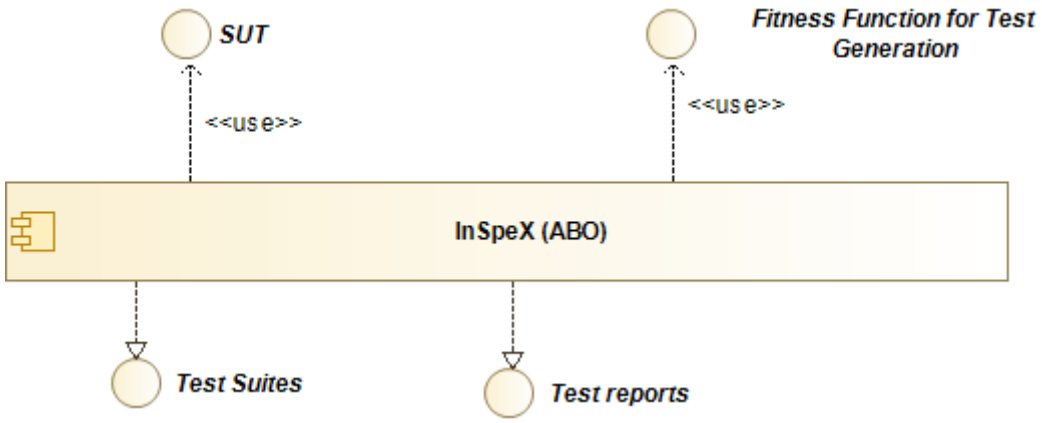

Figure 28 Services and Interfaces

3.6.2.1 Details on realized interfaces

Name	Type
Test Suites	Framework Interfaces and Services
Test reports	Framework Interfaces and Services

Table 49 Realized Interfaces

3.6.2.2 Details on used interfaces

Name	Type
SUT	Framework Interfaces and Services
Fitness Function for Test Generation	InSpeX (ABO)

Table 50 Used Interfaces

3.6.3 Relations to the Framework

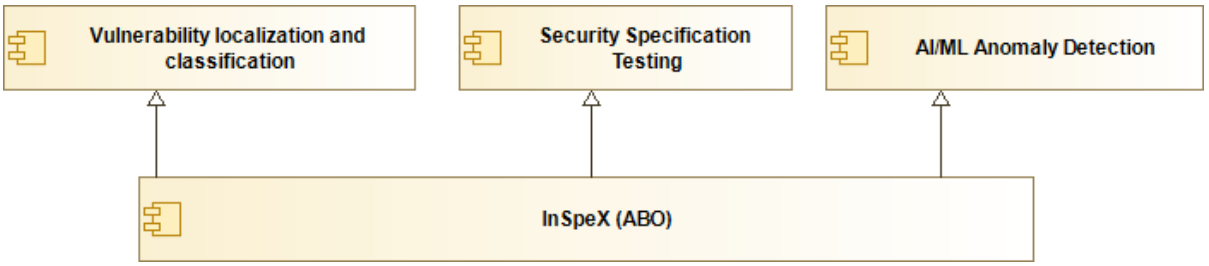

Figure 29 Relations to the Framework

3.6.3.1 Details on realized framework tools

Relation	Framework Tool	Category

The tool will provide preliminary classification and localization of the identified vulnerabilities.	Vulnerability localization and classification	WP4. Prevention at Development
The security specification of the system will be used to construct a fitness function to evaluate the outputs of the system	Security Specification Testing	WP4. Prevention at Development
The tool will use AI/ML techniques to explore the input space and identify anomalies which can be related to vulnerabilities	AI/ML Anomaly Detection	WP3. Reactive Protection at Operations

Table 51 Relations to Framework

3.6.4 Deployment

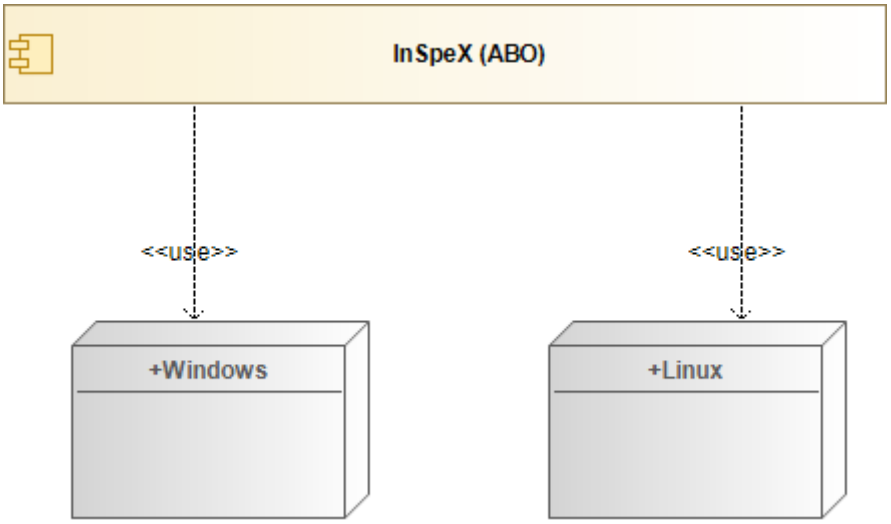

Figure 30 Deployment

3.6.4.1 Deployment details

Usage Type	Node
It will be deployed on Windows.	Windows
It will be deployed on Linux.	Linux

Table 52 Deployment

3.6.5 Interfaces specific to InSpeX (ABO) component

Name	Description
Fitness Function for Test Generation	A fitness function will be used for evaluating if a certain input or sets of inputs lead to a vulnerability. The fitness function can be created to evaluate functional, safety or security aspects.

Table 53 Interfaces specific to InSpeX (ABO) component

3.7 InFuz (ABO)

Given a functional specification of the system under test defined as state model, the tool generates tests by applying different mutation operators at: model level and input level. The tool should provide capabilities for checking that the functional specification satisfies safety and security requirements

3.7.1 Purpose of InFuz (ABO) component

The table below provides the list of the tool features as well as their criticality, planned release date and status.

Properties	Purpose
Criticality: High Release: Final-M32 Status: in progress	ABO-InFuz-010: shall provide modelling of system specification using state based formalisms
Criticality: High Release: Final-M32 Status: in progress	ABO-InFuz-020: shall provide verification of safety and security properties against the specification
Criticality: High Release: Final-M32 Status: in progress	ABO-InFuz-030: shall provide test generation from the model (conformance)
Criticality: High Release: Final-M32 Status: planned	ABO-InFuz-040: shall support generation of invalid inputs from the model (robustness testing)
Criticality: High Release: Final-M32 Status: planned	ABO-InFuz-050: shall provide user friendly test reporting
Criticality: High Release: Final-M32 Status: planned	ABO-InFuz-060: shall support preliminary localization of the source (source, specs, requirements) of the error
Criticality: High Release:	ABO-InFuz-070: shall provide traceability between requirements and generated tests

Final-M32 **Status: planned**

Table 54 InFuz (ABO) component purpose

3.7.2 Services and Interfaces

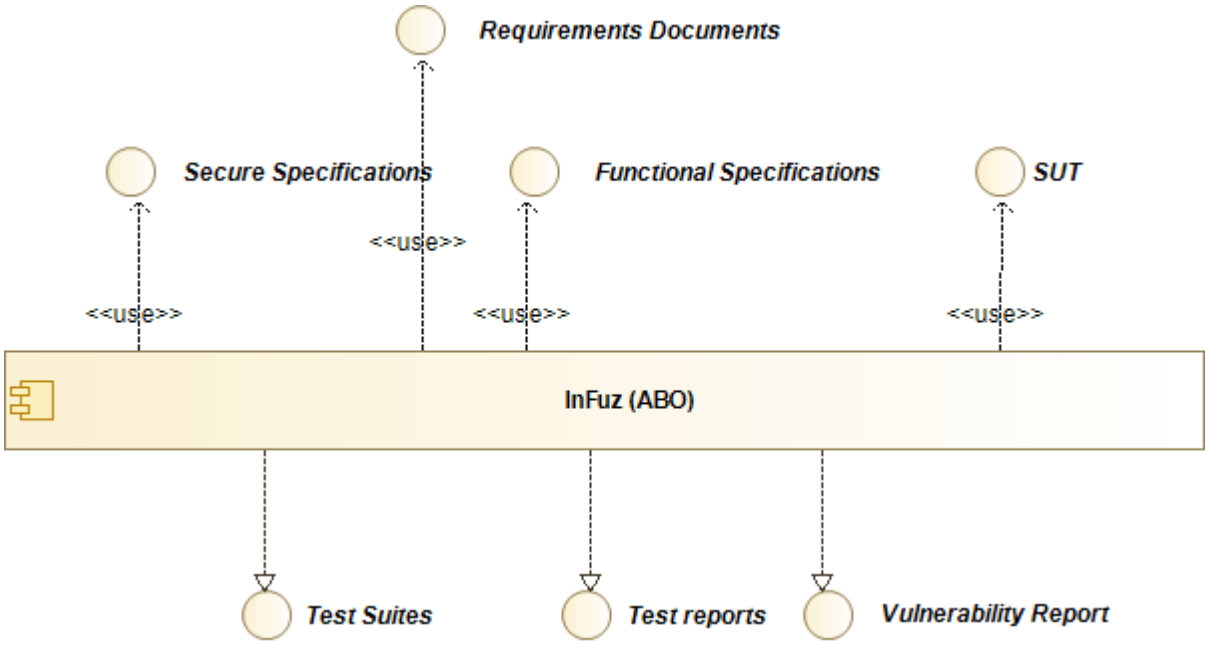

Figure 31 Services and Interfaces

3.7.2.1 Details on realized interfaces

Name	Type
Test Suites	Framework Interfaces and Services
Test reports	Framework Interfaces and Services
Vulnerability Report	InFuz (ABO)

Table 55 Realized Interfaces

3.7.2.2 Details on used interfaces

Name	Type
Secure Specifications	Framework Interfaces and Services
Functional Specifications	Framework Interfaces and Services
SUT	Framework Interfaces and Services
Requirements Documents	Framework Interfaces and Services

Table 56 Used Interfaces

3.7.3 Relations to the Framework

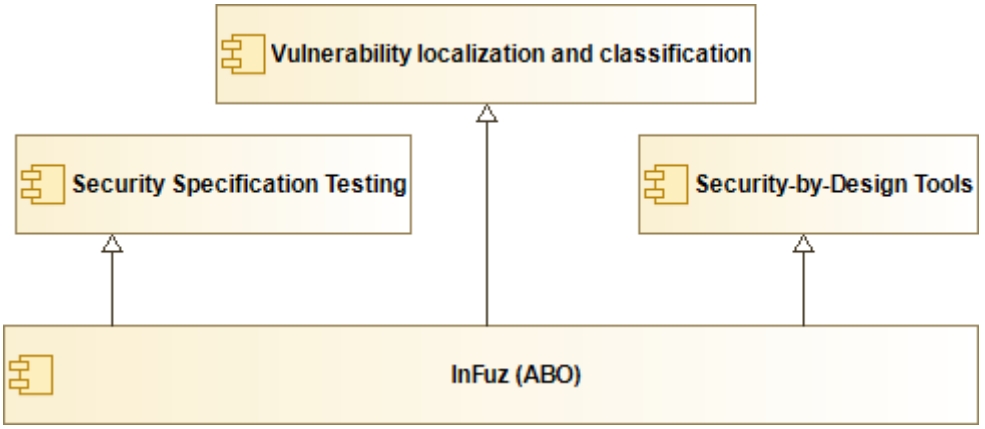

Figure 32 Relations to the Framework

3.7.3.1 Details on realized framework tools

Relation	Framework Tool	Category
The tool will allow to verify if a given functional specification satisfied the security requirements at design level	Security Specification Testing	WP4. Prevention at Development
The tool will provide vulnerability classification and localization	Vulnerability localization and classification	WP4. Prevention at Development
The tool will generate tests from secure-by-by design specifications.	Security-by-Design Tools	WP4. Prevention at Development

Table 57 Relations

3.7.4 Deployment

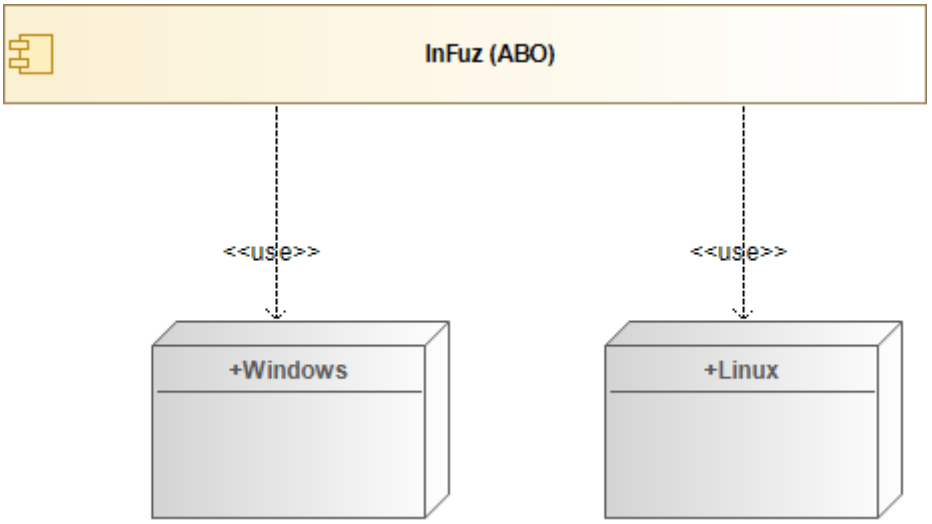

Figure 33 Deployment

3.7.4.1 Deployment details

Usage Type	Node
The tool will be deployed on Windows	Windows
The tool will be deployed on Linux	Linux

Table 58 Deployment

3.7.5 Interfaces specific to InFuz (ABO) component

Name	Description
Vulnerability Report	The tool will generate a report describing potential vulnerabilities it has identified.

Table 59 Interfaces specific to InFuz (ABO) component

3.8 SEAFOX (MDH)

SEAFOX is a tool for test modelling and test generation using combinatorial testing.

3.8.1 Purpose of SEAFOX (MDH) component

The table below provides the list of the tool features as well as their criticality, planned release date and status.

Properties	Purpose
Criticality: Medium **Release: Baseline-M6** **Status: done**	SEAFOX-010: SEAFOX should provide support for CODESYS version 2x

Criticality: High **Release: Baseline-M6** **Status: done**	SEAFOX-020: SEAFOX should provide support for CODESYS version 3x
Criticality: High **Release: Intermediate-M24** **Status: planned**	SEAFOX-030: SEAFOX should support the integration with CODESYS test manager for test script generation
Criticality: High **Release: Final-M32** **Status: planned**	SEAFOX-040: SEAFOX should support the fuzzing of the input parameters

Table 60 SEAFOX (MDH) component purpose

3.8.2 Services and Interfaces

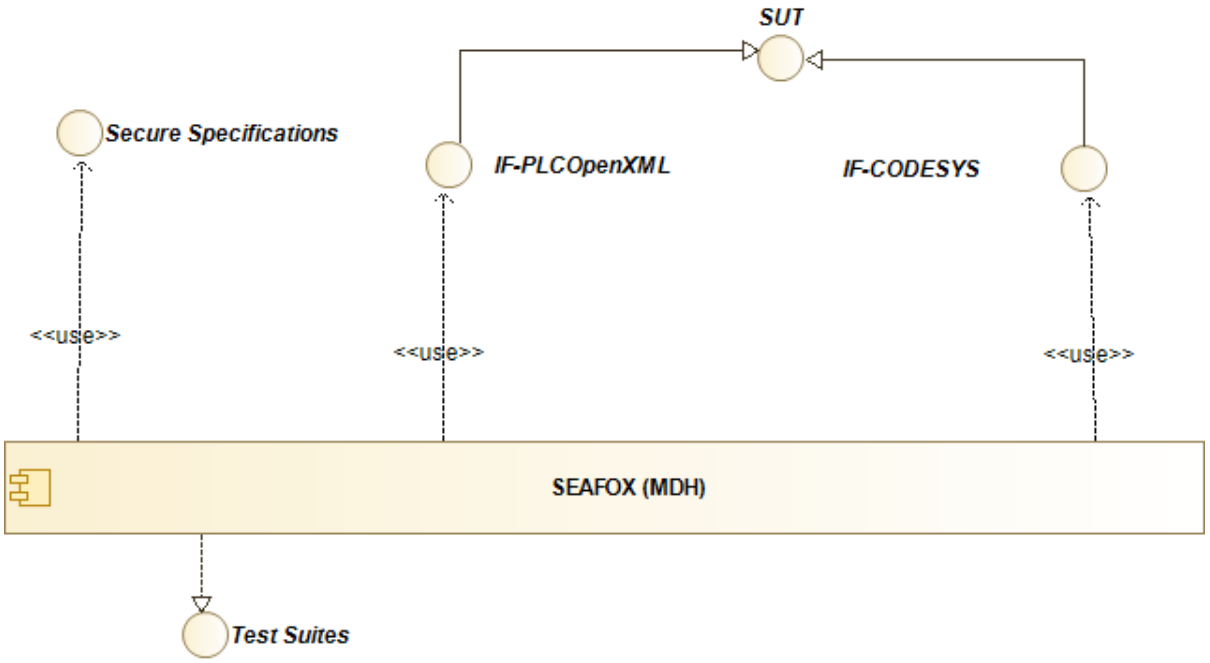

Figure 34 Services and Interfaces

3.8.2.1 Details on realized interfaces

Name	Type
Test Suites	Framework Interfaces and Services

Table 61 Realized Interfaces

3.8.2.2 Details on used interfaces

Name	Type
IF-PLCOpenXML	Framework Interfaces and Services

IF-CODESYS	Framework Interfaces and Services
Secure Specifications	Framework Interfaces and Services

Table 62 Used Interfaces

3.8.2.3 Details on realized framework tools

Relation	Framework Tool	Category
SEAFOX accesses the SUT configuration by addressing the PLC code in OpenXml format.	SUT	Framework Interfaces and Services
SEAFOX needs the code in CODESYS as a part of SUT specification.	SUT	Framework Interfaces and Services

Table 63 Relations

3.8.3 Subordinates

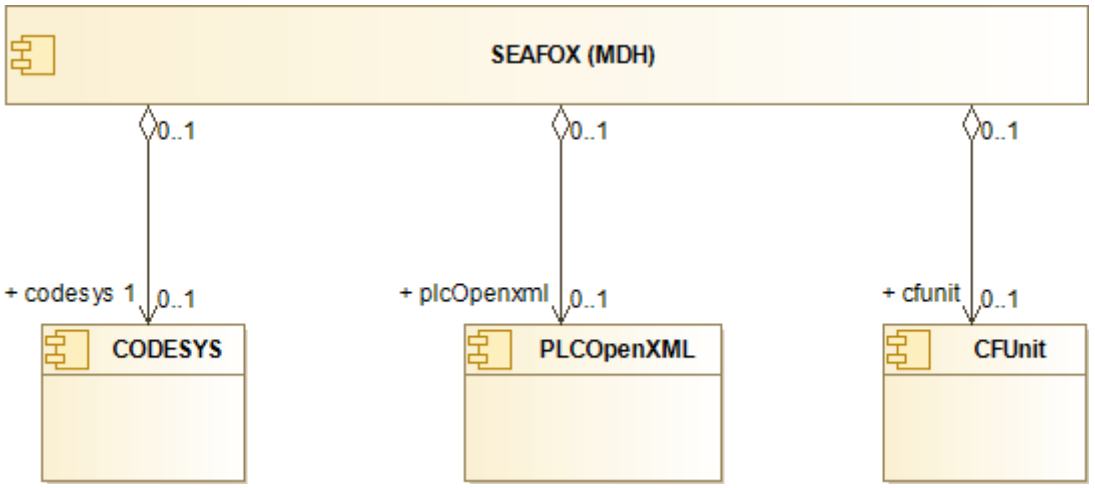

Figure 35 Subordinates

3.8.3.1 Details on subordinate components

Name	Description
CODESYS	CODESYS integration
PLCOpenXML	PLCOpenXML integration
CFUnit	Integration with CFUnit test automation framework

Table 64 Subordinates

3.8.4 Relations to the Framework

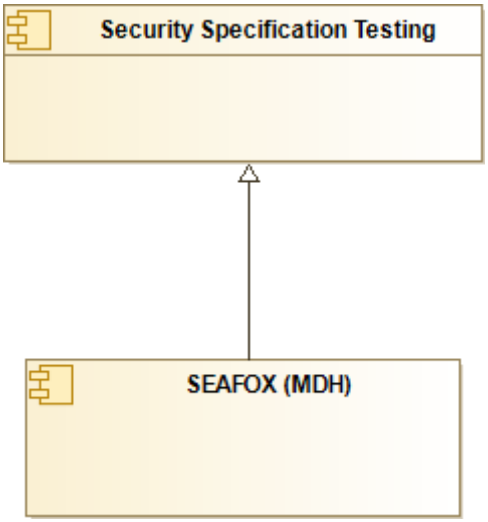

Figure 36 Relations to the Framework

3.8.4.1 Details on realized framework tools

Relation	Framework Tool	Category
SEAFOX implements security specification testing in the context of VeriDevOps framework.	Security Specification Testing	WP4. Prevention at Development

Table 65 Relations to Framework

3.8.5 Deployment

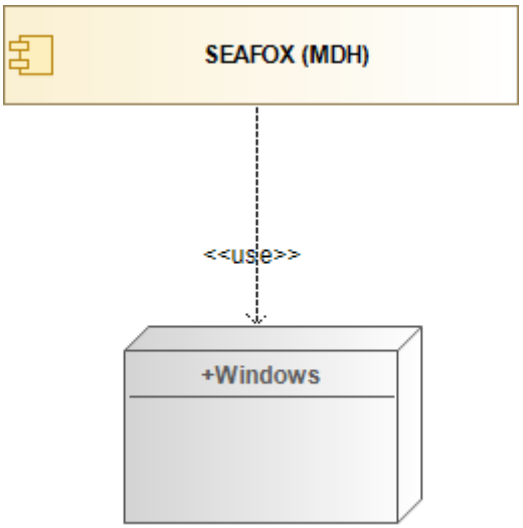

Figure 37 Deployment

3.8.5.1 Deployment details

Usage Type	Node
SEAFOX is a Windows standalone tool.	Windows

Table 66 Deployment

3.9 CompleteTest (MDH)

CompleteTest is a software solution intended to be used for automated testing of Programmable Logic Controllers (PLC) software, found in systems like airplanes, nuclear power plants, medical devices, trains, space shuttles. PLC software is all around us. We use them in our daily life without even considering to what extent their functioning is dependent on software and that they might fail because of a software. And there is one thing that makes these systems rather unique about them: if they fail, people may die, and the environment may be at harm. However, testing these systems is a rather difficult and time-consuming task. There are many national and international standards, safety regulation agencies mandating a certain level of testing. Currently, many companies developing PLC software are manually testing their software which is a tedious and error prone process. CompleteTest is based on the main scientific outcome of different research projects. CompleteTest is rooted in theories of formal verification, model checking, and model-based test generation.

3.9.1 Purpose of CompleteTest (MDH) component

The table below provides the list of the tool features as well as their criticality, planned release date and status.

Properties	Purpose
Criticality: Medium Release: Intermediate-M24 Status: in progress	CT-011: CT should provide support for CODESYS version 2x
Criticality: High Release: Intermediate-M24 Status: planned	CT-021: CT should provide support for CODESYS version 3x
Criticality: High Release: Intermediate-M24 Status: planned	CT-031: CT should support the integration with CODESYS test manager for test script generation
Criticality: High Release: Final-M32 Status: planned	CT-041: CT should support the integration with ProPas

Table 67 CompleteTest (MDH) component purpose

3.9.2 Services and Interfaces

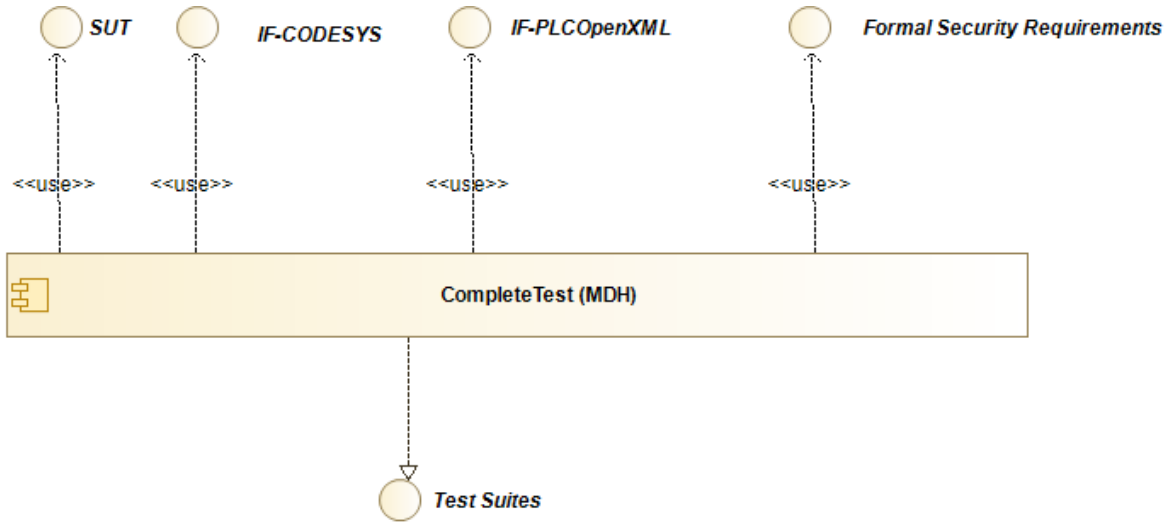

Figure 38 Services and Interfaces

3.9.2.1 Details on realized interfaces

Name	Type
Test Suites	Framework Interfaces and Services

Table 68 Realized Interfaces

3.9.2.2 Details on used interfaces

Name	Type
IF-CODESYS	Framework Interfaces and Services
IF-PLCOpenXML	Framework Interfaces and Services
Formal Security Requirements	Framework Interfaces and Services
SUT	Framework Interfaces and Services

Table 69 Used Interfaces

3.9.3 Subordinates

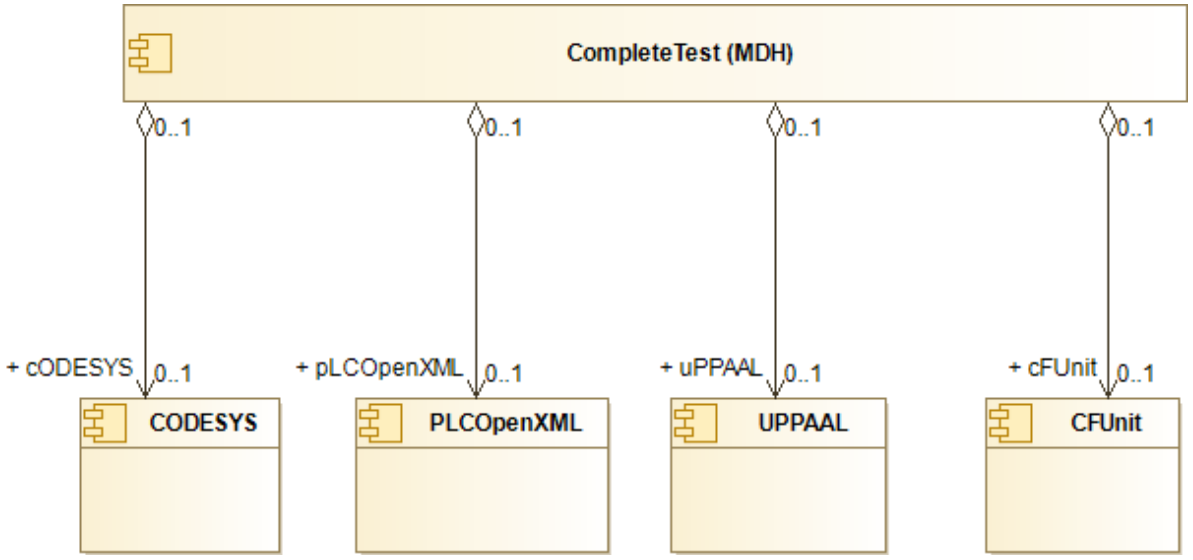

Figure 39 Subordinates

3.9.3.1 Details on subordinate components

Name	Description
CODESYS	CODESYS integration
PLCOpenXML	PLCOpenXML integration
CFUnit	Integration with CFUnit test automation framework
UPPAAL	UPPAAL Model checker used as a test engine

Table 70 Subordinates

3.9.4 Relations to the Framework

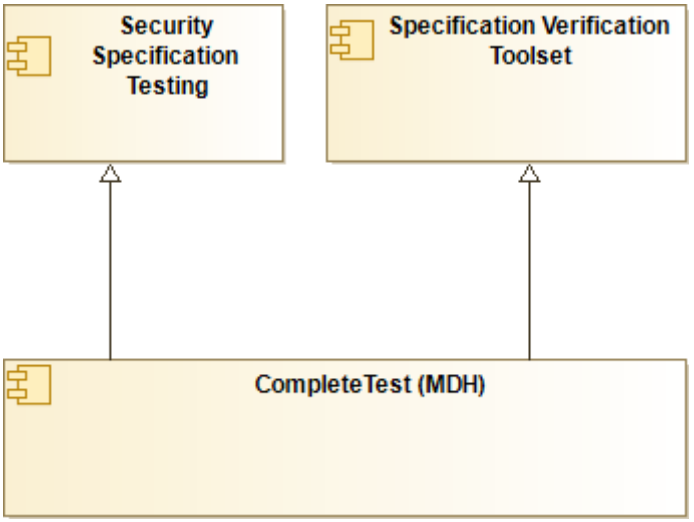

Figure 40 Relations to the Framework

3.9.4.1 Details on realized framework tools

Relation	Framework Tool	Category
CompleteTest is using the functional specification together with the vulnerability property to discover potential security exploits.	Security Specification Testing	WP4. Prevention at Development
CompleteTest includes an automatic model transformation and property generation as part of the verification tool set.	Specification Verification Toolset	WP2. Security Requirements Generation

Table 71 Relations

3.9.5 Deployment

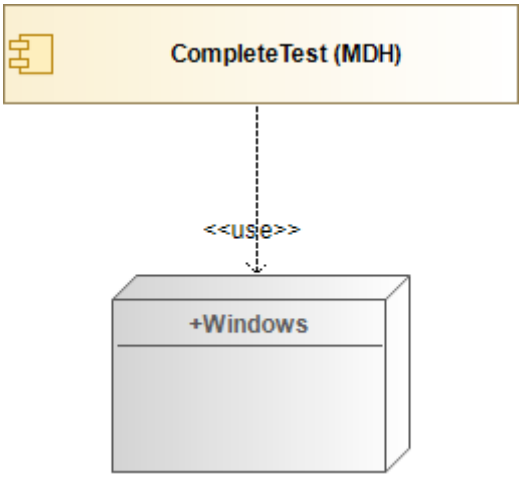

Figure 41 Deployment

3.9.5.1 Deployment details

Usage Type	Node
CompletTest is a Windows standalone tool.	Windows

Table 72 Deployment

3.10 PROPAS (MDH)

PROPAS (The PROperty PAtten Specification and Analysis) is a tool set for automated and formal consistency analysis of industrial critical requirements based on Satisfiability Modulo Theories 1.

3.10.1 Purpose of PROPAS (MDH) component

The table below provides the list of the tool features as well as their criticality, planned release date and status.

Properties	Purpose
Criticality: High **Release:** **Intermediate-M24** **Status: in progress**	PPS-011: security patterns integration with PROPAS
Criticality: High **Release:** **Intermediate-M24** **Status: in progress**	PPS-021: integration with UPPAAL and CompleteTest for design verification

Table 73 PROPAS (MDH) component purpose

3.10.2Services and Interfaces

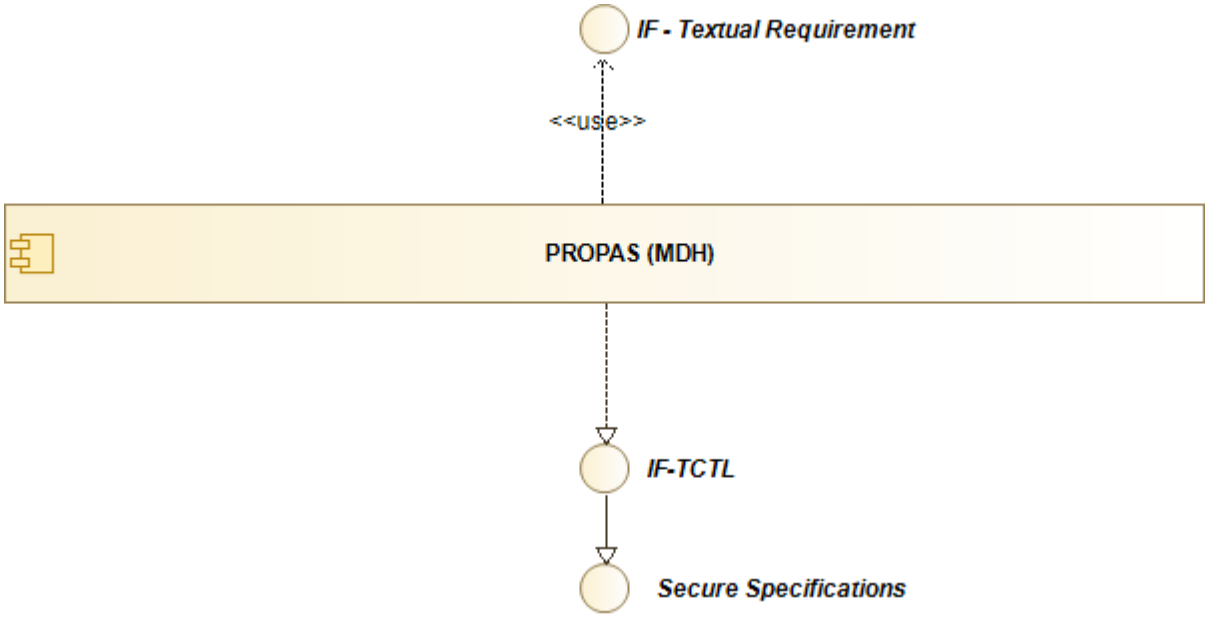

Figure 42 Services and Interfaces

3.10.2.1 Details on realized interfaces

Name	Type
IF-TCTL	PROPAS (MDH)

Table 74 Realized Interfaces

3.10.2.2 Details on used interfaces

Name	Type
IF - Textual Requirement	PROPAS (MDH)

Table 75 Used Interfaces

3.10.2.3 Details on realized framework tools

Relation	Framework Tool	Category
TCTL to be used as a secure specification	Secure Specifications	Framework Interfaces and Services

Table 76 Relations

3.10.3Subordinates

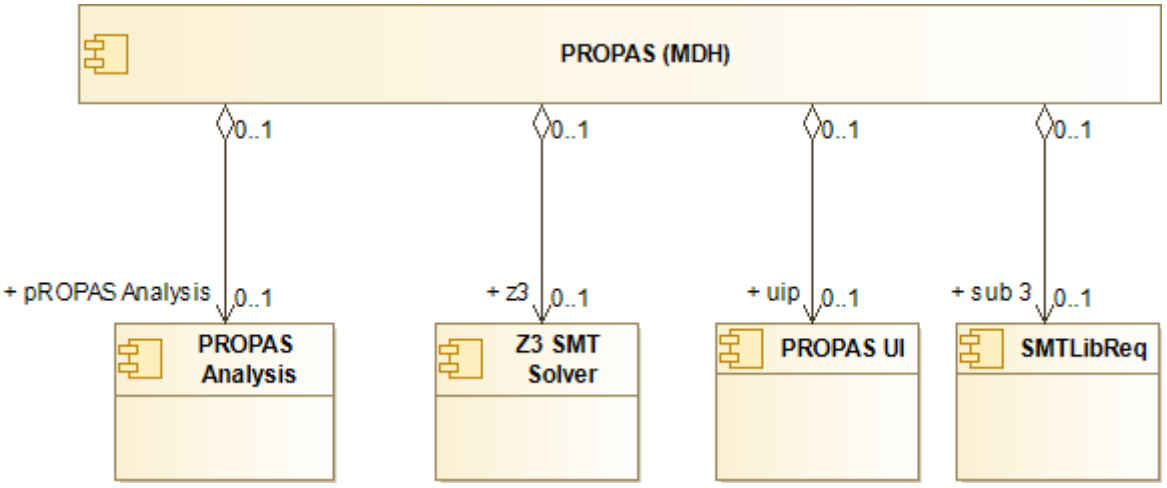

Figure 43 Subordinates

3.10.3.1 Details on subordinate components

Name	Description
Z3 SMT Solver	The current version of the PROPAS tool uses Z3 4 SMT solver and theorem prover from Microsoft Research as consistency checking engine.
PROPAS UI	SMTLibReq is the part of the PROPAS tool which generates an SMT-LIB script suitable for checking based on the formal system specification encoded in TCTL. The parser operates according to the 1:1 principle, meaning that there is a one-to-one mapping between the TCTL properties 2 and the SMT-Lib assertions that constitute the SMT-Lib script 3. The library supports parsing and transformation of arbitrary nested TCTL formulas.
SMTLibReq	SMTLibReq is a generator transforming TCTL to SMT assertions.
PROPAS Analysis	Consistency analysis of FLD requirements using Z3.

Table 77 Subordinates

3.10.4Relations to the Framework

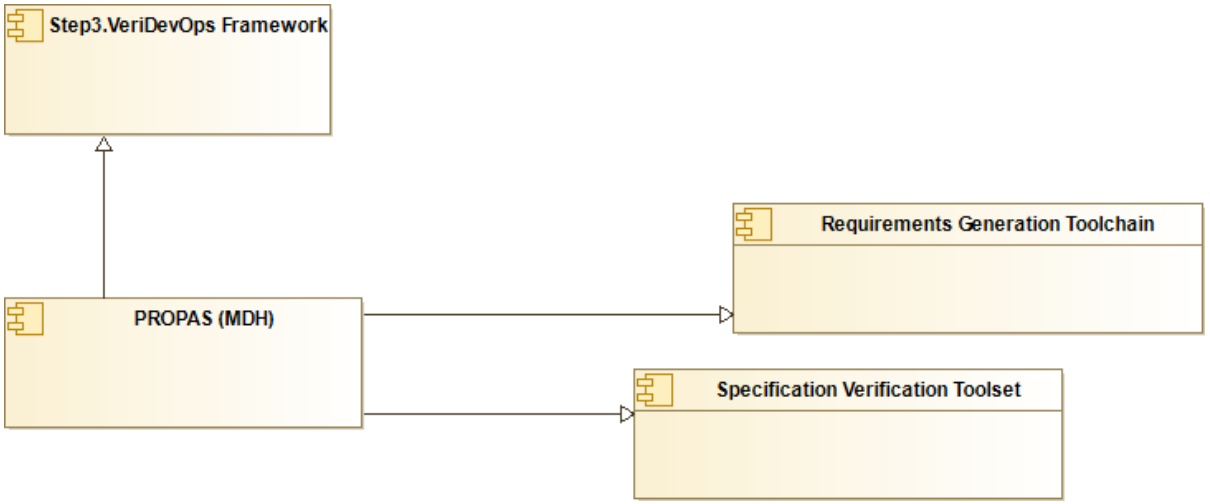

Figure 44 Relations to the Framework

3.10.4.1 Details on realized framework tools

Relation	Framework Tool	Category
PROPAS is a part of the VeriDevOps framework.	Step3.VeriDevOps Framework	VDO Framework
The requirements specification formalized with PROPAS is verified, for example, for consistency.	Specification Verification Toolset	WP2. Security Requirements Generation
PROPAS can be used to generate requirements in pseudo-natural language for further analysis and verification.	Requirements Generation Toolchain	WP2. Security Requirements Generation

Table 78 Relations to Framework

3.10.5 Deployment

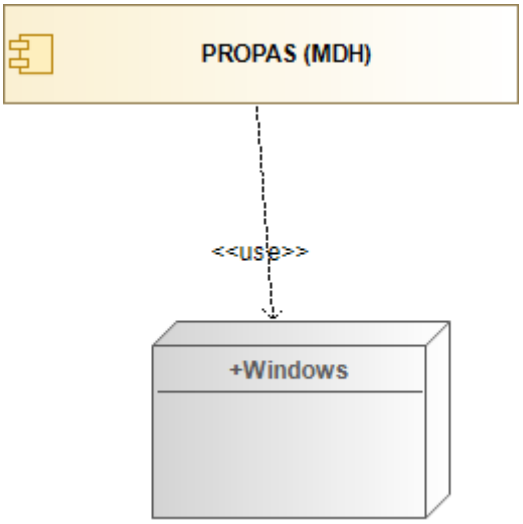

Figure 45 Deployment

3.10.5.1 Deployment details

Usage Type	Node
Tool deployed as a standalone application.	Windows

Table 79 Deployment

3.10.6 Interfaces specific to PROPAS (MDH) component

Name	Description
IF - Textual Requirement	Textual requirement written in natural language
IF-TCTL	Requirement in TCTL format

Table 80 Interfaces specific to PROPAS (MDH) component

3.11 ReSA (MDH)

RESA is an Eclipse-based tool chain for structured requirements specification in ReSA, which scales to multiple architectural levels of abstraction. ReSA is an ontology-based requirements specification language tailored to automotive embedded systems development, which uses requirements boilerplates to structure the specification in natural language. Furthermore, we propose a consistency check function that seamlessly integrates into the tool chain, for the automated consistency check of requirements using Z3 SMT solver.

3.11.1 Purpose of ReSA (MDH) component

The table below provides the list of the tool features as well as their criticality, planned release date and status.

Properties	Purpose
Criticality: High **Release: Intermediate-M24** **Status: planned**	RESA-010: security patterns integration with RESA
Criticality: High **Release: Final-M32** **Status: planned**	RESA-020: integration with UPPAAL and CompleteTest for design verification

Table 81 ReSA (MDH) component purpose

3.11.2 Services and Interfaces

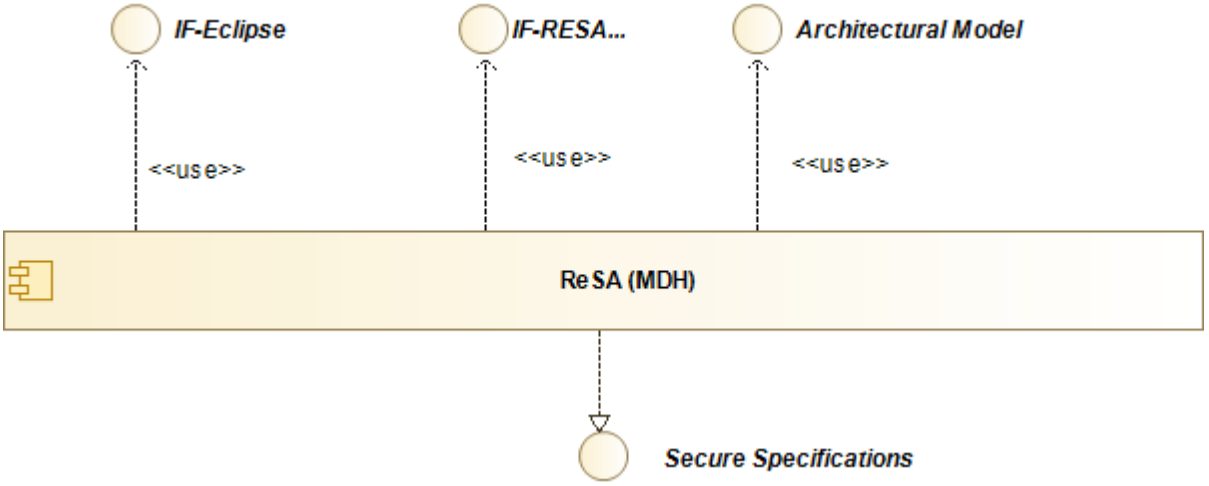

Figure 46 Services and Interfaces

3.11.2.1 Details on realized interfaces

Name	Type
Secure Specifications	Framework Interfaces and Services

Table 82 Realized Interfaces

3.11.2.2 Details on used interfaces

Name	Type
Architectural Model	ReSA (MDH)
IF-RESA File Format	ReSA (MDH)
IF-Eclipse	ReSA (MDH)

Table 83 Used Interfaces

3.11.3Subordinates

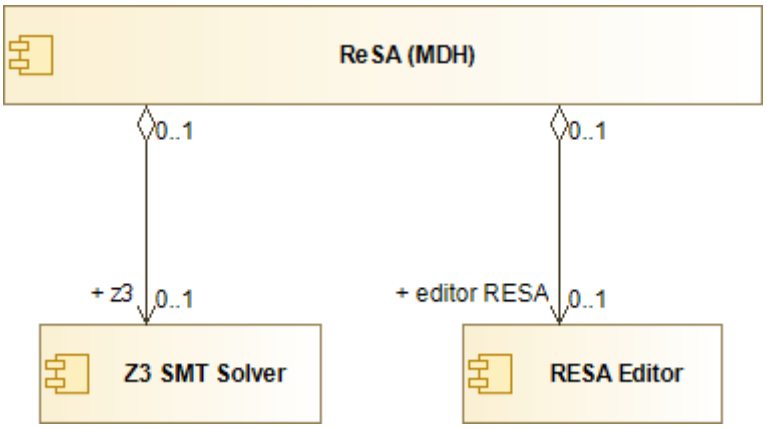

Figure 47 Subordinates

3.11.3.1 Details on subordinate components

Name	Description
Z3 SMT Solver	Z3 is an efficient Satisfiability Modulo Theories (SMT) solver
RESA Editor	Model editor

Table 84 Subordinates

3.11.4Relations to the Framework

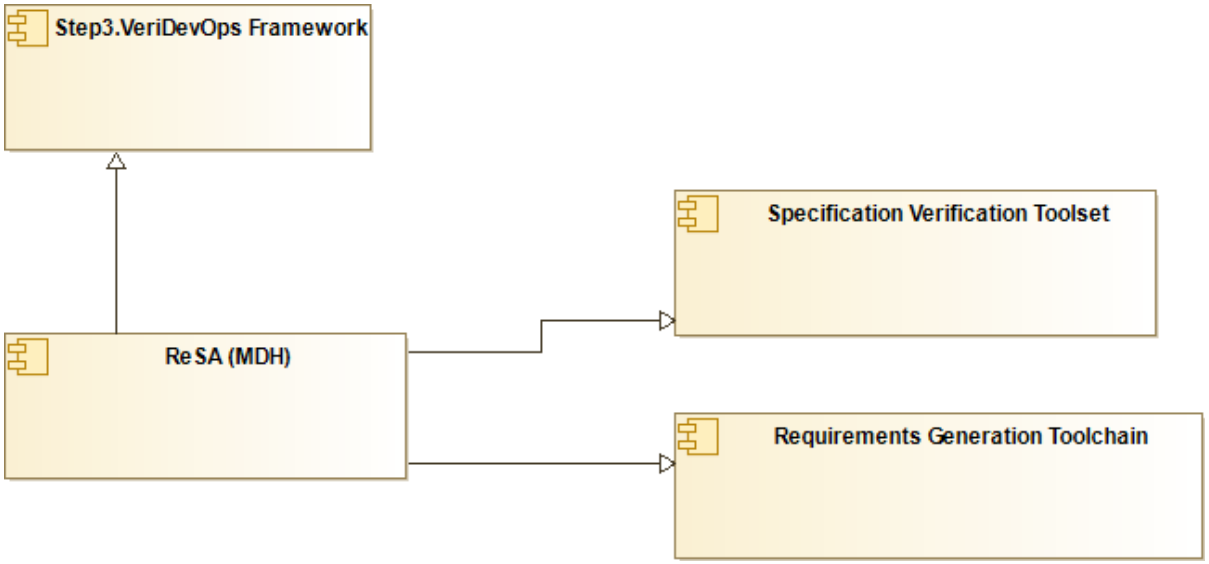

Figure 48 Relations to the Framework

3.11.4.1 Details on realized framework tools

Relation	Framework Tool	Category
ReSA is a part of the VeriDevOps framework.	Step3.VeriDevOps Framework	VDO Framework
ReSA is used for requirements analysis and thus a part of the Specification Verification Toolset by the VeriDevOps framework.	Specification Verification Toolset	WP2. Security Requirements Generation
ReSA can be used for formal requirements generation using boilerplates in pseudo-natural language.	Requirements Generation Toolchain	WP2. Security Requirements Generation

Table 85 Relations to Framework

3.11.5 Deployment

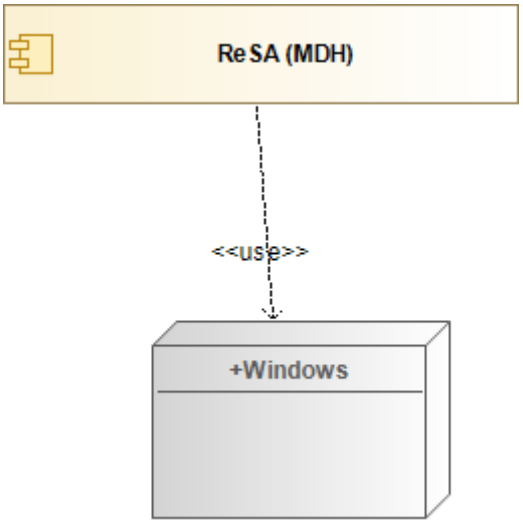

Figure 49 Deployment

3.11.5.1 Deployment details

Usage Type	Node
RESA is deployed as a standalone application.	Windows

Table 86 Deployment

3.11.6 Interfaces specific to ReSA (MDH) component

Name	Description
IF-Eclipse	The tool is using the xText Eclipse Framework. The framework provides an xText editor for grammar specification using the xText grammar language, and generates a start-up IDE based on Eclipse, which includes Parser, Compiler, Linker, and textual editor

IF-RESA Format	**File** Requirements expressed using boiler plates
Architectural Model	Architectural specification given in the EAST-ADL format.

Table 87 Interfaces specific to ReSA (MDH) component

4 Deployment Platforms

This section outlines the deployment platforms for the project tools. This information is provided to inform end-users on any limitations concerning the deployment.

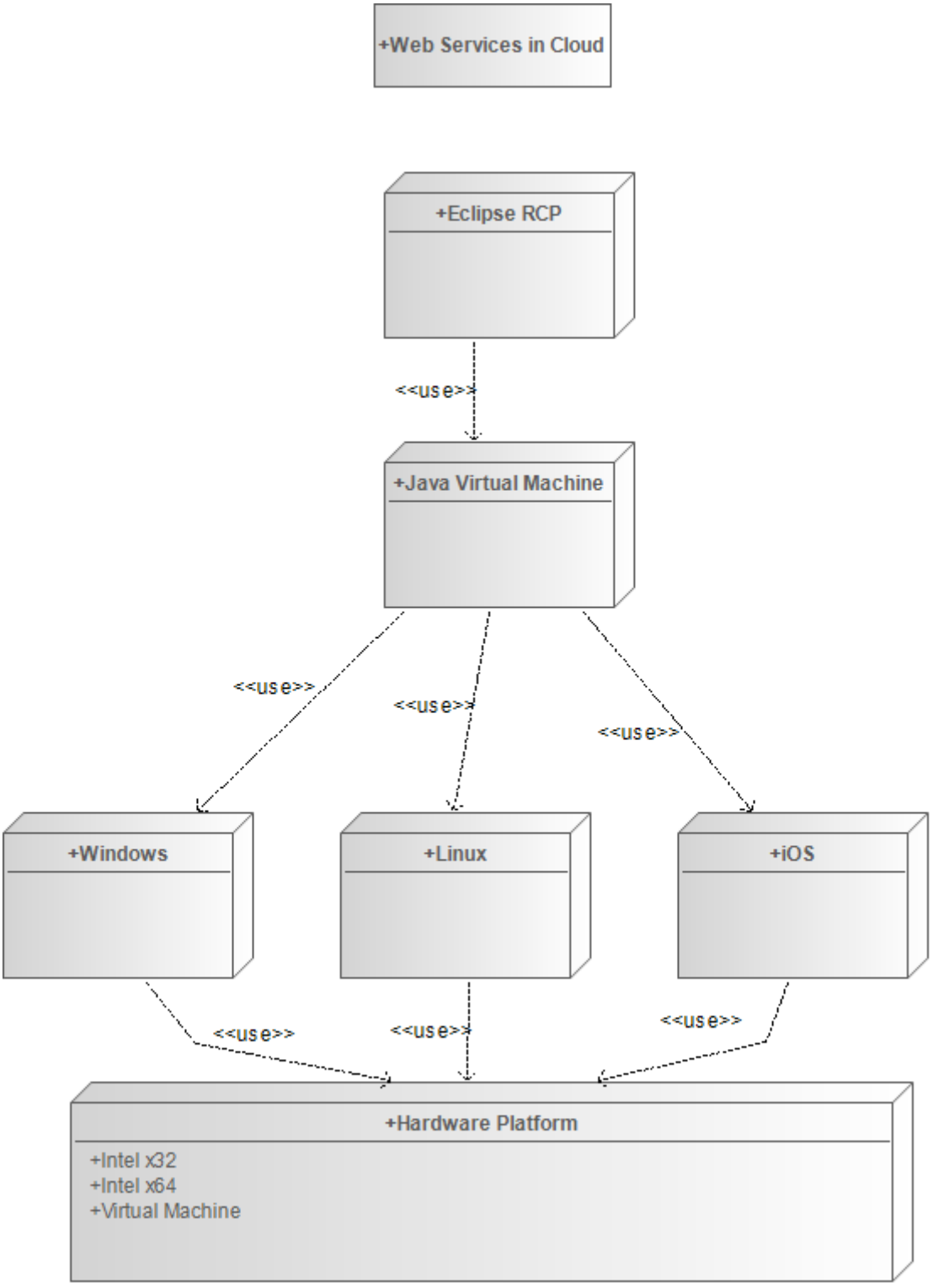

Figure 50 Deployment platforms

Name	Description	Used by
Java Virtual Machine	A Java virtual machine (JVM) is an abstract computing machine that enables a computer to run a Java program.	RQCODE (SOFT), Eclipse RCP
Linux	Linux is free and open-source software operating system distributions built around the Linux kernel.	ARQAN (SOFT), Montimage Monitoring Tool, InSpeX (ABO), InFuz (ABO), Java Virtual Machine
Hardware Platform	Hardware platform for execution.	Montimage Monitoring Tool, Linux, Windows, iOS
Windows	Microsoft Windows is family of graphical operating systems developed, marketed, and sold by Microsoft.	InSpeX (ABO), InFuz (ABO), SEAFOX (MDH), CompleteTest (MDH), PROPAS (MDH), ReSA (MDH), Java Virtual Machine, Tool Component Template (ORG)
iOS	iOS is an operating system created by Apple Inc.	Java Virtual Machine
Eclipse RCP	Eclipse provides the Rich Client Platform (RCP) for developing general purpose applications.	Modelio (SOFT)
Web Services in Cloud	Many tools are deployed as a Web Service and can be accessed over Internet.	ARQAN (SOFT), Montimage Monitoring Tool, THOE (IKER)

Table 88 Platforms

4.1 Hardware Platform

Name	Description	Used by
Intel x32	Intel x32 is the 32-bit version of the x86 instruction set by Intel.	
Intel x64	Intel x64 is the 64-bit version of the x86 instruction set by Intel.	
Virtual Machine	Virtual machine is based on the computer architecture and provides functionality of a physical computer.	

Table 89 Hardware Platform

5 Case Study Requirements Analysis

This section provides a traceability analysis for the requirements by providing a link between case study requirements and tool requirements. This information indicates the coverage for

the case study requirements and responsibilities by the tool providers to address the case studies.

5.1 ABB Case Study Requirements

ID	Definitions	Impacts tools
ABB-010	By using VeriDevOps technologies, requirements for new or enhanced functionality of the standard platform have to be written in such a way that they are maintainable for future extensions but also fit test cases for software module testing and overall regression test.	MODELIO-030, MODELIO-020, RQCODE-010, MMT-010, ABO-InFuz-010, ABO-InFuz-020, SEAFOX-030, CT-041, PPS-011, PPS-021, RESA-010, RESA-020
ABB-020	A VeriDevOps tool chain for requirement specification analysis should support the writing of the requirement but also make them fit for automated testing and selection of test cases/regression tests. This should fit both simulation and on-site commissioning to optimize the efficiency with the test opportunities at hand.	MODELIO-030, MODELIO-110, MODELIO-020, MODELIO-130, MODELIO-140, RQCODE-010, ABO-InSpEx-010, ABO-InSpEx-020, ABO-InSpEx-030, ABO-InFuz-030, ABO-InFuz-040, ABO-InFuz-050, ABO-InFuz-060, SEAFOX-040, CT-041, PPS-021, RESA-010
ABB-030	Requirement models and simulation models produced in VeriDevOps will offer a fast track to efficient, safe, and secure crane operations for both remote and cabin-based crane operation in an authentic environment - as close to real crane operation and realistic conditions as possible.	MODELIO-110, MODELIO-130, MODELIO-140, MMT-010, MMT-020, SEAFOX-040, CT-031, CT-041
ABB-040	A system, preferably a standard, and tool to categorize the customer requirements and link them to internal requirements and systems, could aid a systematic work to determine which requirements should be handled where and to what extent they are fulfilled by the standard or project specific solutions.	MODELIO-010, MODELIO-030, MODELIO-070, MODELIO-020, MODELIO-040, MODELIO-050, MODELIO-060, ARQN-030, ARQN-070, ABO-InFuz-070, SEAFOX-040
ABB-050	Select what to evolve and innovate from assessment of the development process from requirement engineering to regression testing and deployment.	MODELIO-010, MODELIO-030, MODELIO-070, MODELIO-120, MODELIO-020, MODELIO-040, MODELIO-050, MODELIO-060, MODELIO-150, RQCODE-010, ARQN-040, ABO-InSpEx-010, ABO-InSpEx-020, ABO-InSpEx-030, ABO-InFuz-030,

		ABO-InFuz-040, ABO-InFuz-050, ABO-InFuz-060
ABB-060	Improve the current way of writing requirements (both safety, security and the ones used in the regular process).	MODELIO-030, MODELIO-020, RQCODE-010, PPS-011, PPS-021, RESA-020
ABB-070	Improve test case generation, implementation of test cases and the execution of test cases/steps by automating different steps during test design and execution.	MODELIO-110, MODELIO-120, MODELIO-080, MODELIO-130, MODELIO-140, ABO-InSpEx-010, ABO-InSpEx-020, ABO-InSpEx-030, ABO-InFuz-030, ABO-InFuz-040, ABO-InFuz-050, ABO-InFuz-060, SEAFOX-010, SEAFOX-020, SEAFOX-030, CT-011, CT-021
ABB-080	Improve the way vulnerabilities are identified, security aspects are monitored, and traces are analyzed.	MODELIO-010, MODELIO-030, MODELIO-070, MODELIO-110, MODELIO-020, MODELIO-040, MODELIO-050, MODELIO-060, MODELIO-080, MODELIO-130, MODELIO-140, MMT-010, MMT-020, MMT-030, MMT-040, MMT-050, MMT-060, MMT-070, THOE-010, THOE-020, THOE-030, THOE-040, THOE-050, THOE-060, THOE-070
ABB-090	Support nightly builds with automatic regression tests so the engineers can handle bugs and vulnerabilities faster instead of discovering these at release day when the final build is done.	MODELIO-070, MODELIO-110, MODELIO-120, MODELIO-130, MODELIO-150, MODELIO-140, ABO-InSpEx-010, ABO-InSpEx-020, ABO-InSpEx-030, ABO-InFuz-030, ABO-InFuz-040, ABO-InFuz-050, ABO-InFuz-060, SEAFOX-010, SEAFOX-020, SEAFOX-030
ABB-100	Support improving the test practices related to selection and generation of effective and efficient test cases to improve test coverage (i.e., primarily functional testing and secondly module testing).	MODELIO-120, ABO-InSpEx-010, ABO-InSpEx-020, ABO-InSpEx-030, ABO-InFuz-030, ABO-InFuz-040, ABO-InFuz-050, ABO-InFuz-060, SEAFOX-010, SEAFOX-020, SEAFOX-030, CT-011, CT-021
ABB-110	Increase the efficiency in understanding the requirements to be able to implement and test efficiently.	MODELIO-030, MODELIO-020, RQCODE-010, ARQN-030, ARQN-040, ARQN-070, ABO-InFuz-010, ABO-InFuz-020, SEAFOX-010, SEAFOX-020, SEAFOX-030, CT-041

ABB-120	VeriDevOps methods and tools allow for designing a system on several hierarchical layers (e.g. functional and module levels). Artefacts defined on a higher level are to be linked to or be reused on lower levels. Different kinds of views on the system can be designed by using appropriate requirement types.	MODELIO-010, MODELIO-070, RQCODE-010, ABO-InFuz-010, ABO-InFuz-020, CT-041
ABB-130	Validation of design against customer requirements (ensure that the design meets the customer expectations) either for safety or security requirements.	MODELIO-010, MODELIO-030, MODELIO-070, MODELIO-020, MODELIO-040, MODELIO-050, MODELIO-060, MODELIO-090, RQCODE-050, ABO-InFuz-020, CT-041, RESA-010, RESA-020
ABB-140	Generate test case specifications out of requirements models (improve consistency and quality of test cases and accelerate generation of test cases).	MODELIO-030, MODELIO-070, MODELIO-120, MODELIO-020, MODELIO-040, MODELIO-050, MODELIO-060, ABO-InFuz-030, ABO-InFuz-040, CT-011, CT-021, CT-031
ABB-150	Generate test scripts out of test specifications and logic diagrams.	MODELIO-110, MODELIO-120, MODELIO-130, MODELIO-140, SEAFOX-010, SEAFOX-020, SEAFOX-030, CT-011, CT-021, CT-031
ABB-160	Prioritize and select regression test cases according to preset conditions (ensure that relevant test cases are executed at a given time).	MODELIO-140, ABO-InSpEx-010, ABO-InFuz-030, ABO-InFuz-040, CT-011, CT-021, CT-031
ABB-170	Detect security breaches in the ABB system, by relying on different techniques: signature-based monitoring to identify common threats in industrial control systems, and ML/AI algorithms to detect abnormal and suspicious activities that cannot be detected using standards signatures.	MODELIO-110, MODELIO-080, MODELIO-130, MODELIO-140, MMT-010, MMT-020, MMT-030, MMT-040
ABB-180	Determine a list of potential countermeasures that need to be applied to enforce security at runtime.	MODELIO-010, MODELIO-030, MODELIO-070, MODELIO-020, MODELIO-040, MODELIO-050, MODELIO-060, MODELIO-090, MODELIO-150, MMT-060
ABB-190	Analyze ABB system traces at different levels (e.g. network traffic, application logs, etc.) to extract metadata, and adapt a	MODELIO-010, MODELIO-110, MODELIO-080, MODELIO-130, MODELIO-140, MMT-010, MMT-020

	monitoring solution to be able to parse them	
ABB-200	Design security rules that model attacks to be avoided or properties to be achieved.	MODELIO-010, MODELIO-070, MODELIO-090, MMT-020, MMT-030
ABB-210	By configuring or learning the typical behaviour of a particular crane it should be possible to detect if the crane is in a hazardous state and by that avoid an accident.	MODELIO-110, MODELIO-080, MODELIO-130, MODELIO-140, MMT-030, MMT-040
ABB-220	By configuring or learning the typical behaviour of a particular crane it should be possible to detect a security breach or a cyber-attack.	MODELIO-010, MODELIO-110, MODELIO-080, MODELIO-130, MODELIO-140, MMT-010, MMT-020, MMT-030, MMT-040
ABB-230	VeriDevOps methods and devices to increase safety and security by monitoring network communication or devices' internal state.	MODELIO-110, MODELIO-080, MODELIO-130, MODELIO-140, MMT-010, MMT-020, MMT-030, MMT-040, MMT-050, MMT-060, MMT-070, SEAFOX-040
ABB-240	The identification of known vulnerabilities in components and systems is needed. The known vulnerabilities refer to publicly known vulnerabilities in databases, such as, Common Vulnerability Enumeration (CVE) list published by MITRE.	MODELIO-010, MODELIO-030, MODELIO-070, MODELIO-020, MODELIO-040, MODELIO-050, MODELIO-060, MODELIO-080, MODELIO-150, MMT-020, MMT-030, THOE-010, THOE-020, THOE-030, THOE-040, THOE-050, THOE-060, THOE-070, PPS-011, PPS-021, RESA-010, RESA-020
ABB-250	Vulnerabilities identification should cover not only control software, but also software tools for development and maintenance, as well as hardware and device configuration.	MODELIO-010, MODELIO-030, MODELIO-070, MODELIO-020, MODELIO-040, MODELIO-050, MODELIO-060, MODELIO-080, MODELIO-150, MMT-020, MMT-030, THOE-010, THOE-020, THOE-030, THOE-040, THOE-050, THOE-060, THOE-070
ABB-260	VeriDevOps is to be able to monitor known vulnerabilities with the aim of ensuring security in operation.	MODELIO-080, MMT-020, MMT-030, THOE-010, THOE-020, THOE-030, THOE-040, THOE-050, THOE-060, THOE-070

Table 90 ABB Case Study Requirements Analysis

5.2 FAGOR Case Study Requirements

ID	Definitions	Impacts tools
FAG-010	During software development, VeriDevOps outcome will help in testing patches before deploying them to the entire network, then reducing time and effort.	MODELIO-010, MODELIO-030, MODELIO-070, MODELIO-110, MODELIO-120, MODELIO-020, MODELIO-040, MODELIO-050, MODELIO-060, MODELIO-080, MODELIO-090, MODELIO-130, MODELIO-150, MODELIO-140, MMT-060
FAG-020	During machine operation, VeriDevOps will produce automatic detection of known vulnerabilities and attacks, analysis of their root cause and providing effective automatic countermeasures (reaction), or manual recommendations for decision support.	MODELIO-010, MODELIO-030, MODELIO-070, MODELIO-020, MODELIO-040, MODELIO-050, MODELIO-060, MODELIO-090, MODELIO-150, ARQN-050, MMT-010, MMT-020, MMT-030, MMT-040, MMT-050, MMT-060, MMT-070, THOE-010, THOE-020, THOE-030, THOE-040, THOE-050, THOE-060, THOE-070
FAG-030	FAGOR needs an automatic procedure capable of interpreting these security requirements or recommendations such as NIST or STIG, and finding when the device is implementing them and rising alerts when devices do not fulfil them.	MODELIO-010, MODELIO-030, MODELIO-070, MODELIO-020, MODELIO-040, MODELIO-050, MODELIO-060, MODELIO-150, RQCODE-020, RQCODE-040, RQCODE-050, ARQN-040, ARQN-050, MMT-020, MMT-030
FAG-040	FAGOR needs to add to its devices a capability which can be automatic search for vulnerabilities.	MODELIO-030, MODELIO-070, MODELIO-110, MODELIO-040, MODELIO-050, MODELIO-060, MODELIO-080, MODELIO-130, MODELIO-140, THOE-010, THOE-020, THOE-030, THOE-040, THOE-050, THOE-060, THOE-070, ABO-InSpEx-010, ABO-InSpEx-020, ABO-InSpEx-030
FAG-050	FAGOR expects automated requirement interpretation and implementation (WP2)	MODELIO-030, MODELIO-070, MODELIO-110, MODELIO-020, MODELIO-080, MODELIO-130, MODELIO-140, RQCODE-020, RQCODE-040, RQCODE-050, ARQN-030, ARQN-040, ARQN-050, ARQN-070, CT-041, PPS-011, PPS-021, RESA-010, RESA-020
FAG-060	FAGOR expects new testing processes or techniques on software development areas (WP4)	MODELIO-010, MODELIO-070, MODELIO-110, MODELIO-120, MODELIO-090, MODELIO-130, MODELIO-150, MODELIO-140,

		RQCODE-010, ABO-InSpEx-010, ABO-InSpEx-020, ABO-InSpEx-030, ABO-InFuz-030, ABO-InFuz-040, ABO-InFuz-050, ABO-InFuz-060, CT-041, PPS-011, PPS-021, RESA-010, RESA-020
FAG-070	FAGOR expects automated reports about anomalies on cloud environment (WP3)	MMT-070
FAG-080	FAGOR expects vulnerability scanner on edge devices (WP3)	THOE-010, THOE-020, THOE-030, THOE-040, THOE-050, THOE-060, THOE-070
FAG-090	The defined cybersecurity tests should be compliant with international standards and certifications, such as, ISA/IEC 62443 in order to move forward a step on future certification of the IoT platform.	MODELIO-110, MODELIO-120, MODELIO-130, MODELIO-150, MODELIO-140, RQCODE-020, RQCODE-040, RQCODE-050, ARQN-030, ARQN-040
FAG-100	VDO Framework shall provide means for automatic monitoring of the network and service of the MQTT server and receiving alarms or reports if anomalies are detected.	MMT-010, MMT-020, MMT-030, MMT-040
FAG-110	VDO Framework shall provide means to interpret the NIST framework requirements and check in the system if the requirements are implemented or not.	MODELIO-030, MODELIO-070, MODELIO-020, MODELIO-040, MODELIO-050, MODELIO-060, RQCODE-020, RQCODE-040, RQCODE-050, ARQN-030, ARQN-040, ARQN-050, MMT-010, MMT-020, MMT-030, MMT-040
FAG-120	VDO Framework shall provide means to enforce the needed NIST framework requirements and raise notifications about the actions carried out.	MODELIO-010, MODELIO-030, MODELIO-120, MODELIO-020, MODELIO-040, MODELIO-050, MODELIO-060, MODELIO-090, MODELIO-150, RQCODE-020, RQCODE-040, RQCODE-050, MMT-060
FAG-130	VDO Framework shall support with tools a process for identifying vulnerabilities and the needed actions to mitigate them in a repository and crossing those indications with the configuration of the IPC device.	MODELIO-010, MODELIO-030, MODELIO-120, MODELIO-020, MODELIO-040, MODELIO-050, MODELIO-060, MODELIO-090, MODELIO-150, ARQN-030, ARQN-040, ARQN-050, ARQN-070, MMT-060, THOE-010, THOE-020, THOE-030, THOE-040, THOE-050, THOE-060, THOE-070, ABO-InSpEx-010, ABO-InSpEx-020, ABO-InSpEx-030

Table 91 FAGOR Case Study Requirements Analysis

6 VDO Tools Features Roadmap Updates

This section gives a breakdown by the planned release milestone for the tool features. The tables below list the tool feature requirements and give indications on the current status of those features. The tables also provide release notes/comments, if available. In addition, the tables provide traceability information and cross-link the feature requirements with the case study requirements. Thus, the tool providers are able to monitor the impact on the case studies.

6.1 Modelio (SOFT) Features Roadmap

6.1.1 To be available at Baseline-M6 milestone

Feature	Properties	Comments/Release notes	Affected CSR
MODELIO-010: Modelio shall provide system modelling capabilities.	**Criticality:** High **Status:** done	Several system modelling language are already available in Modelio.	*FAG-010*, *FAG-020*, *FAG-030*, *FAG-060*, *FAG-120*, *FAG-130*, *ABB-040*, *ABB-050*, *ABB-080*, *ABB-120*, *ABB-130*, *ABB-180*, *ABB-190*, *ABB-200*, *ABB-220*, *ABB-240*, *ABB-250*
MODELIO-020: Modelio shall provide extendable requirement modelling capabilities.	**Criticality:** High **Status:** done	Extendable requirement engineering support has been implemented in order to support use case scenarii.	*FAG-010*, *FAG-020*, *FAG-030*, *FAG-050*, *FAG-110*, *FAG-120*, *FAG-130*, *ABB-010*, *ABB-020*, *ABB-040*, *ABB-050*, *ABB-060*, *ABB-080*, *ABB-110*, *ABB-130*, *ABB-140*, *ABB-180*, *ABB-240*, *ABB-250*
MODELIO-040: Modelio shall provide checking functionalities.	**Criticality:** High **Status:** done	Audit rules are related to specific standard are available under Modelio	*FAG-010*, *FAG-020*, *FAG-030*, *FAG-040*, *FAG-110*, *FAG-120*, *FAG-130*, *ABB-040*, *ABB-050*, *ABB-080*, *ABB-130*, *ABB-140*,

		ABB-180, *ABB-240*, *ABB-250*
MODELIO-070: Modelio shall manage traceability on project level.	**Modelio Criticality:** High **Status:** done	Modelio supports traceability in related use cases.	*FAG-010*, *FAG-030*, *FAG-020*, *FAG-040*, *FAG-050*, *FAG-060*, *FAG-110*, *ABB-040*, *ABB-050*, *ABB-080*, *ABB-090*, *ABB-120*, *ABB-130*, *ABB-140*, *ABB-180*, *ABB-200*, *ABB-240*, *ABB-250*
MODELIO-090: Modelio shall be able to model attacks on desired system	**Criticality:** High **Status:** done	Modelio supports attack tree model thanks to its customized extension.	*FAG-010*, *FAG-020*, *FAG-060*, *FAG-120*, *FAG-130*, *ABB-130*, *ABB-180*, *ABB-200*
MODELIO-110: Modelio shall provide code generation facilities for specific programming languages (Java, C++, and C for example).	**Criticality:** High **Status:** done	Modelio provides code generation for Java, C++, C#, SQL.	*FAG-010*, *FAG-040*, *FAG-050*, *FAG-060*, *FAG-090*, *ABB-020*, *ABB-030*, *ABB-070*, *ABB-080*, *ABB-090*, *ABB-150*, *ABB-170*, *ABB-190*, *ABB-210*, *ABB-220*, *ABB-230*
MODELIO-120: Modelio shall provide test modelling	**Criticality:** High **Status:** done	UTP profile is already available under Modelio	*FAG-010*, *FAG-060*, *FAG-090*, *FAG-120*, *FAG-130*, *ABB-050*, *ABB-070*, *ABB-100*, *ABB-090*, *ABB-140*, *ABB-150*

Table 92 Features available as baseline by Modelio (SOFT)

6.1.2 To be available at Intermediate-M24 milestone

Feature	Properties	Comments/Release notes	Affected CSR
MODELIO-030: Modelio shall support specific security requirement modelling.	**Criticality:** High **Status:** in progress	Modelio supports UML, SysML and MARTE for functional properties modeling.	*FAG-010*, *FAG-020*, *FAG-030*, *FAG-040*, *FAG-050*, *FAG-110*, *FAG-120*, *FAG-130*, *ABB-010*, *ABB-020*, *ABB-040*, *ABB-050*, *ABB-060*, *ABB-080*, *ABB-110*, *ABB-130*, *ABB-140*, *ABB-180*, *ABB-240*, *ABB-250*

MODELIO-060: Modelio shall provide advanced query functionalities for security purpose.	**Criticality:** High **Status:** in progress	Query functionalities	*FAG-010*, *FAG-020*, *FAG-030*, *FAG-040*, *FAG-110*, *FAG-120*, *FAG-130*, *ABB-040*, *ABB-050*, *ABB-080*, *ABB-130*, *ABB-140*, *ABB-180*, *ABB-240*, *ABB-250*
MODELIO-130: Modelio shall provide modelling support for code generation of security test and analysis.	**Criticality:** Medium **Status:** in progress	Modelio supports several code modelling.	*FAG-010*, *FAG-040*, *FAG-050*, *FAG-060*, *FAG-090*, *ABB-020*, *ABB-030*, *ABB-070*, *ABB-080*, *ABB-090*, *ABB-150*, *ABB-170*, *ABB-190*, *ABB-210*, *ABB-220*, *ABB-230*

Table 93 Features available by Modelio (SOFT) at intermediate release at M24

6.1.3 To be available at Final-M32 milestone

Feature	Properties	Comments/Release notes	Affected CSR
MODELIO-050: Modelio shall provide AI/ML checking functionality and dedicated report.	**Criticality:** High **Status:** planned	Audit rules will be extended to specific standard and use case needs.	*FAG-010*, *FAG-020*, *FAG-030*, *FAG-040*, *FAG-110*, *FAG-120*, *FAG-130*, *ABB-040*, *ABB-050*, *ABB-080*, *ABB-130*, *ABB-140*, *ABB-180*, *ABB-240*, *ABB-250*
MODELIO-080: Modelio shall manage code generation with logger to be able to analyze produced trace during test and modify models accordingly to feedback.	**Criticality:** High **Status:** planned	Modelio will support extendable code generation.	*FAG-010*, *FAG-040*, *FAG-050*, *ABB-070*, *ABB-080*, *ABB-170*, *ABB-190*, *ABB-210*, *ABB-220*, *ABB-230*, *ABB-240*, *ABB-250*, *ABB-260*
MODELIO-140: Modelio shall provide specific code generator for security test and analysis.	**Criticality:** Medium **Status:** in progress	Modelio should provide specific code generation.	*FAG-010*, *FAG-040*, *FAG-050*, *FAG-060*, *FAG-090*, *ABB-020*, *ABB-030*, *ABB-070*, *ABB-080*, *ABB-090*, *ABB-150*, *ABB-160*, *ABB-170*, *ABB-190*, *ABB-210*, *ABB-220*, *ABB-230*

MODELIO-150: Modelio shall provide documentation generation for report production of security analysis. **Criticality:** Medium **Status:** in progress	Modelio should provide specific document generation.	*FAG-010*, *FAG-020*, *FAG-030*, *FAG-060*, *FAG-090*, *FAG-120*, *FAG-130*, *ABB-050*, *ABB-090*, *ABB-180*, *ABB-240*, *ABB-250*

Table 94 Features available by Modelio (SOFT) at final release at M32

6.2 RQCODE (SOFT) Features Roadmap

6.2.1 To be available at Initial-M15 milestone

Feature	Properties	Comments/Release notes	Affected CSR
RQCODE-010: RQCODE shall implement the object-oriented paradigm for requirements specification.	**Criticality:** High **Status:** in progress		*FAG-060*, *ABB-010*, *ABB-020*, *ABB-050*, *ABB-060*, *ABB-110*, *ABB-120*
RQCODE-020: In RQCODE a requirement is represented as a class.	**Criticality:** High **Status:** in progress		*FAG-030*, *FAG-050*, *FAG-090*, *FAG-110*, *FAG-120*
RQCODE-030: RQCODE patterns should incapsulate textual description of requirements with enforcement and verification means.	**Criticality:** High **Status:** in progress		

Table 95 Features available by RQCODE (SOFT) at initial release at M15

6.2.2 To be available at Intermediate-M24 milestone

Feature	Properties	Comments/Release notes	Affected CSR
RQCODE-040: RQCODE shall propose a ready to use catalogue of object-oriented requirements patterns.	**Criticality:** High **Status:** planned		*FAG-030*, *FAG-050*, *FAG-090*, *FAG-110*, *FAG-120*
RQCODE-050: RQCODE shall be extensible to adapt to various enforcement and verification mechanisms such as tests or windows shell scripts.	**Criticality:** High **Status:** planned		*FAG-030*, *FAG-050*, *FAG-090*, *FAG-110*, *FAG-120*, *ABB-130*

Table 96 Features available by RQCODE (SOFT) at intermediate release at M24

6.3 ARQAN (SOFT) Features Roadmap

6.3.1 To be available at Initial-M15 milestone

Feature	Properties	Comments/Release notes	Affected CSR
ARQN-010: The tool shall process in the PDF format.	**Criticality:** High **Status:** done		
ARQN-020: The tool shall extract security requirements from documents in PDF.	**Criticality:** High **Status:** in progress		

Table 97 Features available by ARQAN (SOFT) at initial release at M15

6.3.2 To be available at Intermediate-M24 milestone

Feature	Properties	Comments/Release notes	Affected CSR
ARQN-030: The tool shall categorize requirements according to the standard categories of security requirements.	**Criticality:** Medium **Status:** in progress		*ABB-040*, *ABB-110*, *FAG-090*, *FAG-110*, *FAG-050*, *FAG-130*
ARQN-040: The tool shall propose relevant recommendations to prevent security vulnerabilities based on STIG guidelines.	**Criticality:** Medium **Status:** planned		*ABB-050*, *ABB-110*, *FAG-090*, *FAG-030*, *FAG-110*, *FAG-050*, *FAG-130*

Table 98 Features available by ARQAN (SOFT) at intermediate release at M24

6.3.3 To be available at Final-M32 milestone

Feature	Properties	Comments/Release notes	Affected CSR
ARQN-050: ARQAN shall provide User Interface for navigating lists of classified security requirements, and their mappings to recommendations and patterns.	**Criticality:** Medium **Status:** planned		*FAG-020*, *FAG-030*, *FAG-110*, *FAG-050*, *FAG-130*
ARQN-060: ARQAN shall provide User Interface for classification pipeline administration.	**Criticality:** Medium		

	Status: planned		
ARQN-070: ARQAN shall provide patterns matching for implementing mappings of security requirements in English with patterns of formal security requirements.	**Criticality:** Medium **Status:** planned		*ABB-040*, *ABB-110*, *FAG-050*, *FAG-130*

Table 99 Features available by ARQAN (SOFT) at final release at M32

6.4 THOE(IKER) Features Roadmap

6.4.1 To be available at Baseline-M6 milestone

Feature	Properties	Comments/Release notes	Affected CSR
THOE-010: THOE shall use NVD (National Vulnerability Database) as the main source for searching publicly-known vulnerabilities. https://nvd.nist.gov/	**Criticality:** High **Status:** done		*FAG-080*, *ABB-080*, *FAG-020*, *FAG-040*, *FAG-130*, *ABB-240*, *ABB-250*, *ABB-260*

Table 100 Features available as baseline by THOE(IKER)

6.4.2 To be available at Initial-M15 milestone

Feature	Properties	Comments/Release notes	Affected CSR
THOE-020: THOE shall search for vulnerabilities in NVD by means of hardware and software Common Platform Enumeration identifiers, that is, it shall provide query functionalities given a certain CPE identifier	**Criticality:** High **Status:** in progress		*FAG-080*, *ABB-080*, *FAG-020*, *FAG-040*, *FAG-130*, *ABB-240*, *ABB-250*, *ABB-260*
THOE-030: THOE shall enable the visualization of vulnerabilities	**Criticality:** High **Status:** in progress		*FAG-080*, *ABB-080*, *FAG-020*, *FAG-040*, *FAG-130*, *ABB-240*, *ABB-250*, *ABB-260*

Table 101 Features available by THOE(IKER) at initial release at M15

6.4.3 To be available at Intermediate-M24 milestone

Feature	Properties	Comments/Release notes	Affected CSR
THOE-040: THOE shall generate a report with the list of vulnerabilities and corresponding CVSS values that measures the vulnerability severity	**Criticality:** High **Status:** planned		*FAG-080*, *ABB-080*, *FAG-020*, *FAG-040*, *FAG-130*, *ABB-240*, *ABB-250*, *ABB-260*
THOE-050: THOE shall support the import of CPE-based asset inventory	**Criticality:** High **Status:** planned		*FAG-080*, *ABB-080*, *FAG-020*, *FAG-040*, *FAG-130*, *ABB-240*, *ABB-250*, *ABB-260*

Table 102 Features available by THOE(IKER) at intermediate release at M24

6.4.4 To be available at Final-M32 milestone

Feature	Properties	Comments/Release notes	Affected CSR
THOE-060: THOE shall support the visualization and download of the logs information	**Criticality:** Medium **Status:** planned		*FAG-080*, *ABB-080*, *FAG-020*, *FAG-040*, *FAG-130*, *ABB-240*, *ABB-250*, *ABB-260*
THOE-070: THOE shall enable the configuration of a scheduler for searching vulnerabilities for a given product	**Criticality:** Low **Status:** planned		*FAG-080*, *ABB-080*, *FAG-020*, *FAG-040*, *FAG-130*, *ABB-240*, *ABB-250*, *ABB-260*

Table 103 Features available by THOE(IKER) at final release at M32

6.5 Montimage Monitoring Tool Features Roadmap

6.5.1 To be available at Baseline-M6 milestone

Feature	Properties	Comments/Release notes	Affected CSR
MMT-010: Solution to monitor network traffic. It allows to analyse more than 600 different protocols and to provide a deep insight of the network usage, performance and security.	**Criticality:** High **Status:** done		*ABB-010*, *FAG-020*, *FAG-100*, *FAG-110*, *ABB-030*, *ABB-080*, *ABB-170*, *ABB-190*, *ABB-220*, *ABB-230*

Table 104 Features available as baseline by Montimage Monitoring Tool

6.5.2 To be available at Initial-M15 milestone

Feature	Properties	Comments/Release notes	Affected CSR
MMT-020: Security monitoring module allows to retrieve relevant security attributes from system and application traces in the context of VeriDevOps case studies. The security module relies on a plugin architecture to add new types of traces in the MMT monitoring solution.	**Criticality:** High **Status:** in progress	ABB network traces as well as CSV logs are already integrated in the MMT solution at M9.	*FAG-020*, *FAG-030*, *FAG-100*, *FAG-110*, *ABB-030*, *ABB-080*, *ABB-170*, *ABB-190*, *ABB-200*, *ABB-220*, *ABB-230*, *ABB-240*, *ABB-250*, *ABB-260*
MMT-030: The rule-based analysis module relies on adequate security properties that are specified in XML and inspired from LTL.	**Criticality:** High **Status:** in progress	Several rules are already specified for ABB case study at M9. They are under internal assessment.	*FAG-020*, *FAG-030*, *FAG-100*, *FAG-110*, *ABB-080*, *ABB-170*, *ABB-200*, *ABB-210*, *ABB-220*, *ABB-230*, *ABB-240*, *ABB-250*, *ABB-260*
MMT-040: The ML/AI analysis module allow to detect anomalies of case study traces. This also target encrypted traffic analysis that is relevant for secure communications.	**Criticality:** High **Status:** in progress	A first clustering algorithm is being tested on ABB case study... A study of encrypted traffic analysis is in progress.	*FAG-020*, *FAG-100*, *FAG-110*, *ABB-080*, *ABB-170*, *ABB-210*, *ABB-220*, *ABB-230*

Table 105 Features available by Montimage Monitoring Tool at initial release at M15

6.5.3 To be available at Intermediate-M24 milestone

Feature	Properties	Comments/Release notes	Affected CSR
MMT-050: The root cause analysis module allows to determine the origin of security incidents.	**Criticality:** Medium		*FAG-020*, *ABB-080*, *ABB-230*

	Status: planned		
MMT-060: A list of recommendations are proposed to mitigate the risk after a security incident. This list a part of a resilience catalogue to be defined during the project.	**Criticality:** Medium **Status:** planned		*FAG-010*, *FAG-020*, *FAG-120*, *FAG-130*, *ABB-080*, *ABB-180*, *ABB-230*

Table 106 Features available by Montimage Monitoring Tool at intermediate release at M24

6.5.4 To be available at Final-M32 milestone

Feature	Properties	Comments/Release notes	Affected CSR
MMT-070: Integration of all the previous modules with a friendly user interface.	**Criticality:** High **Status:** planned		*FAG-020*, *FAG-070*, *ABB-080*, *ABB-230*

Table 107 Features available by Montimage Monitoring Tool at final release at M32

6.6 InSpeX (ABO) Features Roadmap

6.6.1 To be available at Final-M32 milestone

Feature	Properties	Comments/Release notes	Affected CSR
ABO-InSpEx-010: shall provide automated and targeted functional, safety and security test generation	**Criticality:** High **Status:** in progress		*ABB-020*, *ABB-050*, *ABB-070*, *ABB-090*, *ABB-100*, *ABB-160*, *FAG-040*, *FAG-060*, *FAG-130*
ABO-InSpEx-020: shall provide human friendly test reports	**Criticality:** Medium **Status:** planned		*ABB-020*, *ABB-050*, *ABB-070*, *ABB-090*, *ABB-100*, *FAG-040*, *FAG-060*, *FAG-130*
ABO-InSpEx-030: shall provide machine-readable test scripts	**Criticality:** Medium **Status:** planned		*ABB-020*, *ABB-050*, *ABB-070*, *ABB-090*, *ABB-100*, *FAG-040*, *FAG-060*, *FAG-130*

Table 108 Features available by InSpeX (ABO) at final release at M32

6.7 PROPAS (MDH) Features Roadmap

6.7.1 To be available at Intermediate-M24 milestone

Feature	Properties	Comments/Release notes	Affected CSR
PPS-011: security patterns integration with PROPAS	**Criticality:** High **Status:** in progress		*ABB-010*, *ABB-060*, *ABB-240*, *FAG-050*, *FAG-060*
PPS-021: integration with UPPAAL and CompleteTest for design verification	**Criticality:** High **Status:** in progress		*ABB-010*, *ABB-020*, *ABB-060*, *ABB-240*, *FAG-050*, *FAG-060*

Table 109 Features available by PROPAS (MDH) at intermediate release at M24

6.8 CompleteTest (MDH) Features Roadmap

6.8.1 To be available at Intermediate-M24 milestone

Feature	Properties	Comments/Release notes	Affected CSR
CT-011: CT should provide support for CODESYS version 2x	**Criticality:** Medium **Status:** in progress		*ABB-070*, *ABB-100*, *ABB-140*, *ABB-150*, *ABB-160*
CT-021: CT should provide support for CODESYS version 3x	**Criticality:** High **Status:** planned		*ABB-070*, *ABB-100*, *ABB-140*, *ABB-150*, *ABB-160*
CT-031: CT should support the integration with CODESYS test manager for test script generation	**Criticality:** High **Status:** planned		*ABB-140*, *ABB-150*, *ABB-160*, *ABB-030*

Table 110 Features available by CompleteTest (MDH) at intermediate release at M24

6.8.2 To be available at Final-M32 milestone

Feature	Properties	Comments/Release notes	Affected CSR
CT-041: CT should support the integration with ProPas	**Criticality:** High		*ABB-110*, *ABB-120*, *ABB-130*, *ABB-010*, *ABB-020*, *ABB-030*, *FAG-050*, *FAG-060*

	Status: planned		

Table 111 Features available by CompleteTest (MDH) at final release at M32

6.9 InFuZ (ABO) Features Roadmap

6.9.1 To be available at Final-M32 milestone

Feature	Properties	Comments/Release notes	Affected CSR
ABO-InFuz-010: shall provide modelling of system specification using state based formalisms	**Criticality:** High **Status:** in progress		*ABB-120*, *ABB-010*, *ABB-110*
ABO-InFuz-020: shall provide verification of safety and security properties against the specification	**Criticality:** High **Status:** in progress		*ABB-130*, *ABB-010*, *ABB-110*, *ABB-120*
ABO-InFuz-030: shall provide test generation from the model (conformance)	**Criticality:** High **Status:** in progress		*ABB-020*, *FAG-060*, *ABB-050*, *ABB-070*, *ABB-090*, *ABB-100*, *ABB-140*, *ABB-160*
ABO-InFuz-040: shall support generation of invalid inputs from the model (robustness testing)	**Criticality:** High **Status:** planned		*ABB-020*, *FAG-060*, *ABB-050*, *ABB-070*, *ABB-090*, *ABB-100*, *ABB-140*, *ABB-160*
ABO-InFuz-050: shall provide user friendly test reporting	**Criticality:** High **Status:** planned		*ABB-020*, *FAG-060*, *ABB-050*, *ABB-070*, *ABB-090*, *ABB-100*
ABO-InFuz-060: shall support preliminary localization of the source (source, specs, requirements) of the error	**Criticality:** High **Status:** planned		*ABB-020*, *FAG-060*, *ABB-050*, *ABB-070*, *ABB-090*, *ABB-100*
ABO-InFuz-070: shall provide traceability between requirements and generated tests	**Criticality:** High **Status:** planned		*ABB-040*

Table 112 Features available by InFuZ (ABO) at final release at M32

6.10 SEAFOX (MDH) Features Roadmap

6.10.1 To be available at Baseline-M6 milestone

Feature	Properties	Comments/Release notes	Affected CSR
SEAFOX-010: SEAFOX should provide support for CODESYS version 2x	**Criticality:** Medium **Status:** done		*ABB-070*, *ABB-090*, *ABB-100*, *ABB-110*, *ABB-150*
SEAFOX-020: SEAFOX should provide support for CODESYS version 3x	**Criticality:** High **Status:** done		*ABB-070*, *ABB-090*, *ABB-100*, *ABB-110*, *ABB-150*

Table 113 Features available as baseline by SEAFOX (MDH)

6.10.2 To be available at Intermediate-M24 milestone

Feature	Properties	Comments/Release notes	Affected CSR
SEAFOX-030: SEAFOX should support the integration with CODESYS test manager for test script generation	**Criticality:** High **Status:** planned		*ABB-070*, *ABB-090*, *ABB-100*, *ABB-110*, *ABB-150*, *ABB-010*

Table 114 Features available by SEAFOX (MDH) at intermediate release at M24

6.10.3 To be available at Final-M32 milestone

Feature	Properties	Comments/Release notes	Affected CSR
SEAFOX-040: SEAFOX should support the fuzzing of the input parameters	**Criticality:** High **Status:** planned		*ABB-230*, *ABB-020*, *ABB-030*, *ABB-040*

Table 115 Features available by SEAFOX (MDH) at final release at M32

6.11 RESA (MDH) Features Roadmap

6.11.1 To be available at Intermediate-M24 milestone

Feature	Properties	Comments/Release notes	Affected CSR

RESA-010: security patterns integration with RESA	**Criticality:** High **Status:** planned		*ABB-010*, *ABB-020*, *ABB-130*, *ABB-240*, *FAG-050*, *FAG-060*

Table 116 Features available by RESA (MDH) at intermediate release at M24

6.11.2 To be available at Final-M32 milestone

Feature	Properties	Comments/Release notes	Affected CSR
RESA-020: integration with UPPAAL and CompleteTest for design verification	**Criticality:** High **Status:** planned		*ABB-010*, *ABB-060*, *ABB-130*, *ABB-240*, *FAG-050*, *FAG-060*

Table 117 Features available by RESA (MDH) at final release at M32

Summary

This document presented the architecture of the VeriDevOps framework and its constituent parts. In addition, it summarised the requirements analysis by providing traceability of the case study requirements and tool feature requirements. The document provides a roadmap as a list of tool features to be available at various milestones related to the case study development and evaluation. This document and the model from which it was generated will be used as reference throughout the project, but will be updated on the fly whenever necessary.