VeriDevOps

Automated Protection and Prevention to Meet Security Requirements in DevOps Environments

D5.1 Report on the architecture and implementation evaluation - initial version

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:	31/3/2022
Author(s):	IKER, MDU
Partners contributed:	ALL Partners in VeriDevOps
Release date:	30th of March 2022
Version	01
Revision:	1
Abstract:	This deliverable describes the integration plan for VeriDevOps framework, and defines the evaluation plan for framework and toolset.
Status:	PU (Public)

Table of Contents

Executive Abstract

The objective of this deliverable is to provide an overview of the VeriDevOps architecture along with its components and to define a preliminary evaluation plan. An integration roadmap is defined for the integration of the software stack's components developed within the project in the VeriDevOps framework. KPIs for the evaluation of each tool are defined, as well as KPIs for the use case scenarios. An evaluation plan is defined in order to evaluate individual components, as well as an initial plan on the integration needs within the framework, and how these are checked against the project objectives and requirements.

The two evaluation iterative phases as well as the continuous validation by case providers are strengthening the collaboration between case studies providers and technology providers. Final evaluation results of the architecture and implementation will be shown in the final version of this deliverable, *D5.2 Report on the architecture and implementation evaluation – final version*.

1. Introduction

The VeriDevOps project assumes a DevOps development process and provides a set of interconnected tools for security requirements generation, prevention and security at design and development, and reactive protection at operations. The set of features provided by the VeriDevOps framework will be validated by the industrial partners ABB and FAGOR, which act as the end-users, through two case studies.

The present document is the first deliverable result of Task 5.1 "Architecture and implementation evaluation" belonging to WP5 "Evaluation and Validation". The purpose of the task is to evaluate the architectural approach defined in WP1[1], as well as the components developed in WP2, WP3 and WP4. The tools will be evaluated individually, using tool-specific KPIs. However, since the individual capability of each tool may be limited compared to the potential of several tools working together, the integration of components in a common framework is also to be evaluated in this task. For that purpose, project objective-related KPIs will be used, as well as KPIs defined for case studies.

At the time of writing this deliverable, most of the tools that compose the VeriDevOps framework are still under development. Therefore, the aim of this initial version of the deliverable is not to provide the complete evaluation of the architecture and components but to establish an integrated approach for the framework, and an evaluation plan for the tools themselves as the integrated framework in whole defining KPIs that will enable us to compare the results with the objectives of the project presented in "B1.1.2 Targeted Results of VeriDevOps" through an empirical study.

The evaluation results for the selected tools will also be presented in the final version of this deliverable. The final architecture and implementation evaluation will be presented in the final version of this deliverable "D5.2 Report on the architecture and implementation evaluation – final version" due in month 32.

The current deliverable D5.1 is divided into the following sections:

- VeriDevOps Architecture Overview, in which we present an overview of the overall architecture of the VeriDevOps framework.

[1] D1.4 VeriDevOps Framework Architecture and Roadmap

- VeriDevOps Tool Components, where we give an overview of the tools that are being developed in the project and specify how they will be evaluated with tool-specific KPIs.
- Integration Roadmap, where we present an integrated approach for putting different tools together through their APIs.
- Evaluation Plan, where a plan is presented to evaluate the framework and tools through the case study scenarios.

2. VeriDevOps Architecture Overview

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

The VeriDevOps toolsets 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 complementary to a certain extent. The case studies will employ a mix of those tools that better correspond to the requirements and will fit better with the methodology or industry practice.

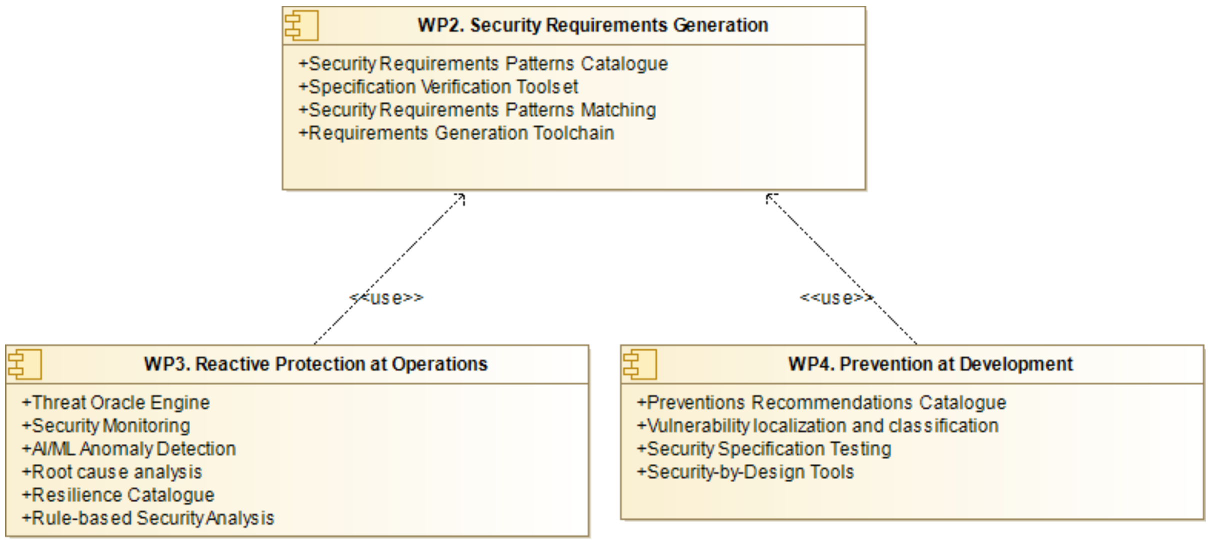
WP2. Security Requirements Generation
+Security Requirements Patterns Catalogue
+Specification Verification Toolset
+Security Requirements Patterns Matching
+Requirements Generation Toolchain
<<use>>
<<use>>
WP3. Reactive Protection at Operations
+Threat Oracle Engine
+Security Monitoring
+AI/ML Anomaly Detection
+Root cause analysis
+Resilience Catalogue
+Rule-based Security Analysis
WP4. Prevention at Development
+Preventions Recommendations Catalogue
+Vulnerability localization and classification
+Security Specification Testing
+Security-by-Design Tools

Figure 2.1 Framework Architecture Overview.

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

When dealing with tools integration, it is crucial to evaluate the multiple concerns related to the coupling of the VeriDevops framework. The coupling level can cross from "tightly-coupled" to a "loosely-coupled" framework across several levels. In a "tightly-coupled" strategy, the inner configuration of an element is directly related to another one so that a difference in this part involves the structure of the other one. In a "loosely-coupled" system, the interaction is restricted to exchanging or transferring data.

Concentrating on the VeriDevops toolset chaining and framework integration, the following list outlines the significant characteristics of coupling tools:

- **Technology Dependency** occurs when communicating systems are bound to standard technology from lower platform requirements (e.g., Windows vs Mac, .NET vs. Java VM) to the higher-level aspects related to integration and interoperability mechanisms. D1.4 illustrates the natural clustering of VeriDevOps tools based on their technological background.
- **Data Format Dependency** is how data elements, exchanged between tools, are defined. This characteristic is essential for VeriDevOps framework integration to exchange data and sample artifacts without losing information.

3. VeriDevOps Tool Components

This section contains the lool description extracted or extended from other deliverables (D.1.4) as well as the KPI definitions.

3.1. Modelio (SOFT) - WP2

Modelio (SOFT): is an open-source modeling 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 (for security), and to combine them with a rich graphical interface for dedicated diagrams, model element property editors and action command controls.

3.2. RQCODE (SOFT) - WP2

The RQCODE is a Java-based SDK to implement the formalization of security 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.

3.3. ARQAN (SOFT) - WP2

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.

3.4. Montimage Monitoring Tool (MI) - WP3

Montimage Monitoring Tool (MMT) is a monitoring solution that allows capturing and analyzing network traffic. It can be used to understand how a 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 analyze system and application logs provided by industrial systems, to check a set of 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 a security incident and anomalies will allow one to better select the countermeasure that can be applied among a set of potential resilience catalogs.

3.5. THOE (IKER) - WP3

The Threat Oracle Engine is an active publicly known vulnerability searching tool for discovering vulnerabilities of a system, using as a source the information from online international vulnerability databases, such as, for example, an NVD[2] database. Fast and

[2] https://nvd.nist.gov/

effective detection of known vulnerabilities is a must for the prevention of cyberattacks. In that sense, THOE will allow industrial control systems to have:

- Automated publicly known vulnerability detection during the whole lifecycle of a product.
- Continuous monitoring for vulnerabilities.
- Searching for vulnerabilities from different online data sources through a single API.

Automation in vulnerability detection is the main benefit offered by the THOE tool, saving a lot of time and resources on being aware of new vulnerabilities and checking them against the system. However, for this automation to be effective, it is essential to have an exhaustive inventory of the assets that need to be monitored and to have every item mapped to its CPE[3] (Common Platform Enumeration) reference. The generation of these inventories can be really hard work, especially if done manually. Thus, by automating this task, THOE will provide a great benefit, especially in the form of time and cost savings.

3.6. InSpeX (ABO) - WP4

InSpeX is a tool that generates tests efficiently for systems with large input spaces in a black-box manner. InSpeX uses a combination of metaheuristic and machine learning approaches to optimize the test generation by focusing on the input regions perceived to be more error-prone or as possible sources of vulnerabilities.

3.7. EaRLY (ABO) - WP3

EaRLY (Early intrusion and anomaly detection) is a toolset for early intrusion and anomaly detection of security attacks by monitoring network traffic. The main feature of the tool is that it is able to detect in real-time, with a certain probability, ongoing network attacks before they are completed.

3.8. InFuz (ABO) - WP4

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

[3] https://nvd.nist.gov/products/cpe

3.9. SEAFOX (MDH) - WP4

SEAFOX is a tool for test modeling and test generation using combinatorial testing for PLC programs and integrated with the CODESYS environment. The tool is focusing on both functional and security testing and the input model is constructed automatically using an OpenPLC XML file.

3.10. CompleteTest (MDH) - WP4

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. CompleteTest is rooted in theories of formal verification, model checking, and model-based test generation and can be used for finding vulnerabilities at the code level.

3.11. PROPAS (MDH) - WP2

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

3.12. ReSA (MDH) - WP2

RESA is an Eclipse-based toolchain 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 toolchain, for the automated consistency check of requirements using Z3 SMT solver.

3.13. NALABS (MDH) - WP2

Based on several natural language smells, NALABS established a set of indicators for requirement flaws and defined dictionary-based metrics to automatically detect these smells in natural language artifacts. NALABS is a new tool that aims at the quick analysis of requirements by detecting problematic requirements (both functional and security-related requirements).

3.14. Tool Specific KPIs - Initial Version

In the following table, TBM stands for To Be Measured, TBE is To Be Estimated, TBI stands for To be Implemented, N/A is Not Applicable and X/Y stands for case study specific values.

TOOL NAME	KPIs Descriptions	Initial Value (e.g., Baseline if exists or can be estimated)	Current Values (to be measured in 2022)	Targeted Values	Measurement Method
NALABS	- Requirement Issues Found; - Requirement Issues Density,	- 10 issues per 300 requirements (3,3% problematic requirements detected)	TBM	10 issues found (per 100 requirements)	Measured using a controlled experiment and quantitative data collection
SEAFOX	- time to generate test cases, test coverage	15 min on average per SUT, 90% code coverage	TBM	10 min on average per SUT, 100% code coverage	Measured using a controlled experiment and quantitative data collection
MMT-SEC	Accuracy	Only classical IP based network - 0% for case study contexts	TBM	more than 90%	Measuring by comparing detected attacks v.s. known attacks
MMT-SEC	Detection time	less than 1 sec	TBM	less than 1 sec	Measuring the difference between timestamps (attack and alert)
MMT-ML	Accuracy	N/A	TBM	more than 90%	Measuring by comparing detected anomalies v.s. known anomalies

MMT-ML	Detection time	N/A	TBM	less than 3 min	Measuring the difference between timestamps (attack and alert)
MMT-RCA	Accuracy	N/A	TBM	more than 95%	Measuring by comparing detected causes v.s. known causes
MMT-RCA	Response time	N/A	TBM	less than 5 sec	Measuring the difference between timestamps (alert and cause)
MMT-RCA	Similarity score	N/A	TBM	more than 90%	The value is provided in the MMT-RCA dashboard
inFuz	Fault detection rate	X/Y	TBM	> 20%	Measured using a controlled experiment and quantitative data collection
inFuz	No gen tests	N/A	TBM	TBE	Measured using a controlled experiment and quantitative data collection
inFuz	FDR*Coverage	N/A	TBM	TBE	Measured using a controlled experiment and quantitative data collection
inFuz	test generation time	N/A	TBM	TBE	Measured using a controlled experiment and quantitative data collection

EaRLY	Attack Detection Rate	N/A	0.756	more than 90%	Measured using a controlled experiment and quantitative data collection
EaRLY	Early detection rate	N/A	0.838	more than 90%	Measured using a controlled experiment and quantitative data collection
EaRLY	Average detection time	N/A	0.04 milliseconds per packet	≤ 0.01 milliseconds per packet	Measured using a controlled experiment and quantitative data collection
EaRLY	Balanced Accuracy	N/A	0.803	more than 90%	Measured using a controlled experiment and quantitative data collection
EaRLY	Precision	N/A	0.703	more than 90%	Measured using a controlled experiment and quantitative data collection
InSpeX	Fault detection rate	X/Y	TBM	> 20%	Measured using a controlled experiment and quantitative data collection
InSpex	No gen tests	N/A	TBM	TBE	Measured using a controlled experiment and quantitative data collection
InSpex	FDR*Coverage	N/A	TBM	TBE	Measured using a

					controlled experiment and quantitative data collection
InSpex	test generation time	N/A	TBM	TBE	Measured using a controlled experiment and quantitative data collection
THOE	Number of supported CPEs	TBE	TBM	>95%	Measured using a controlled experiment
THOE	CPE mapping rate	TBE	TBI	>95%	Measured using a controlled experiment
THOE	Vulnerability notification	TBE	TBI	TBE	Measured using a controlled experiment
ARQAN	Requirements Extraction	N/A	F1=0.8	F1=0.86	F1 measure for predicting whether a sentence is a requirement.
ARQAN	Sec Requirements Extraction	N/A	F1=0.7	F1=0.8	F1 measure for predicting whether a requirements sentence is related to security.
ARQAN	Semantic Search	N/A	N/A	Accuracy of 0.8	Based on subjective evaluation of search results on the STIGs dataset
RQCODE	Number of security patterns in repository	N/A	10	100	Requirements patterns from temporal constrains and

					various guidelines including STIG and OWASP.
RQCODE	Level of reuse for security requirements	N/A	N/A	1.5	Average Number of security findings (e.g. STIGs) per Number of implementations in RQCODE patterns.

Table 3.1 Tool specific KPIs: KPIs Descriptions, Initial Value, Current Values, Targeted Values, Measurement Method.

4. Integration Roadmap

The deliverable *D1.4 VeriDevOps Architecture and Roadmap* proposed an architecture for the VeriDevOps framework, as well as use case-related requirements that it should fulfill in each of the phases. The aim of this section is to propose a roadmap for the integration of the tools that compose the VeriDevOps Framework, while in the final version of this deliverable *D5.2 Report on the architecture and implementation evaluation - final version*, the final architecture of the framework will be validated and evaluated against the expectations and needs of the use case providers.

The main technical result of the VeriDevOps project is the VeriDevOps Framework, a framework that groups the tools presented in the previous section for Security Requirements Generation, Reactive Protection at Runtime as well as for Prevention at Design and Development. These tools need to be highly interconnected (Figure 4.1) in order to achieve the goal of linking security requirements with design analysis, verification at the code level and the runtime analysis of systems.

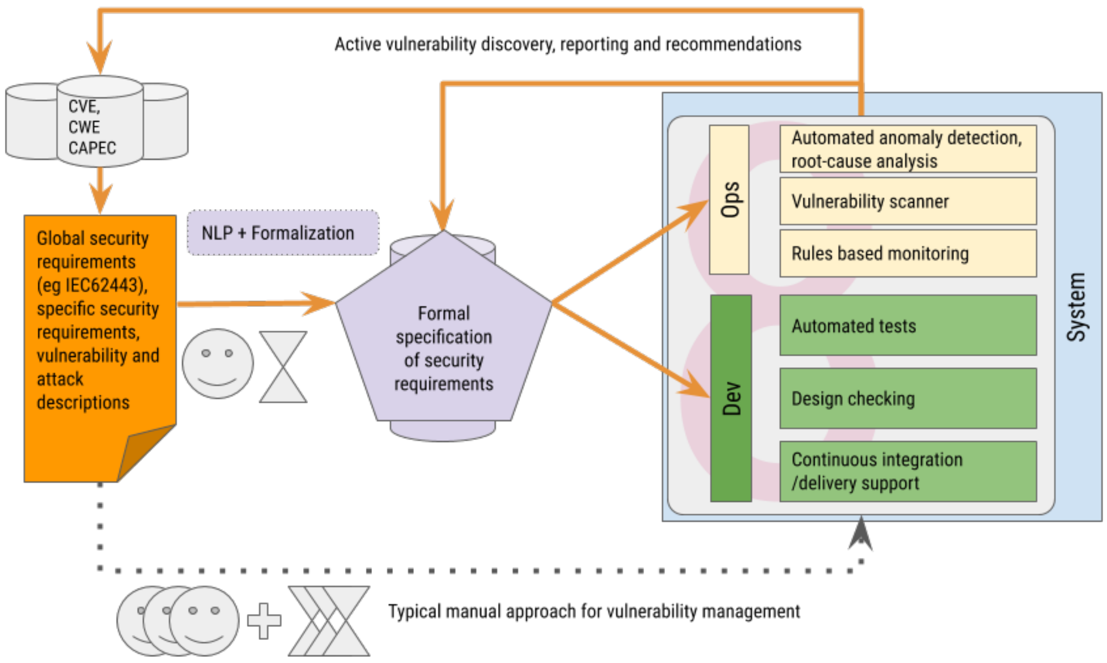
Active vulnerability discovery, reporting and recommendations
CVE, CWE CAPEC
Global security requirements (eg IEC62443), specific security requirements, vulnerability and attack descriptions
NLP + Formalization
Formal specification of security requirements
Ops
Automated anomaly detection, root-cause analysis
Vulnerability scanner
Rules based monitoring
Dev
Automated tests
Design checking
Continuous integration /delivery support
System
Typical manual approach for vulnerability management

Figure 4.1 VeriDevOps Overall Architecture Framework

While the tools provided by different partners may differ in matters such as licensing policies or maturity, every one of them should implement certain functionality and interconnections with the rest of the tools, and enable the configuration of different combinations of tools depending on the needs of the use case. Thus, the framework should be customizable and the tools interchangeable. Whether the benefits of using the tools in combination will be higher compared to using them individually is something that will also be evaluated during the evaluation process.

4.1. Interoperability capabilities

The interoperability capabilities of the framework will be determined by the interfaces of each of the tools. Also, the deployment constraints imposed by each of the tools will determine the compatible baseline for the delivery of the integrated framework. Thus, with the objective of integrating the individual tools into the VeriDevOps Framework and setting a roadmap for the toolset and framework release schedule, it is necessary to identify and adopt a suitable common packaging and delivery approach, as well as define the adequate integration approach, taking into account the supported tool interfaces and their deployment constraints.

The integration approach will be determined by the coupling level of the separate components. The coupling level can span between “tightly-coupled” to “loosely-coupled”. In the former, components are directly related to each other so that the change in one of them affects the structure of the other. In the latter, the only interaction is done by sharing or exchanging data. The possible coupling capacity of different components is largely determined by their dependencies, such as technology dependencies (eg. Linux vs. Windows, .NET vs Java), or data format dependencies. These are defined during 2022 and will be presented in the final version of this deliverable.

4.2. Integration Process

Some tool providers in the VeriDevOps project provide more than one tool to the framework. The first phase of the integration process will consist of each partner integrating its own tools, while in the second phase the tools provided by different partners will be put together. A tightly-coupling approach may be possible between tools from the same provider since they may have been developed with the same dependencies and high integration level in mind from the very beginning stages. However, a tightly-coupled strategy is not valid for the whole framework integration with tools coming from multiple providers, and with a diversity of dependencies. Thus, a loosely-coupled approach should be chosen for the integration of the VeriDevOps Framework.

The tools from the VeriDevOps framework have originated from a variety of technological baselines with the objective to satisfy the needs of a wide variety of diverse industrial domains. Consequently, a flexible and adaptable integration approach is needed, where each use case owner will be able to implement a case-specific integration solution by selecting and configuring a specific set of tools from those included in the framework.

The loosely-coupled approach will allow the users to select the tools they want to use based on their specific goals and own criteria. Thus, the user will only select the tools from the framework that are needed for their use case, and do not have to install the entire framework. By providing a formal interface for input and output data exchange, the tools will be independent, reusable, and interchangeable.

The loosely-coupled approach will also help to ensure the extensibility of the framework, allowing future inclusion of new tools during the project and beyond. The aim of having an extensible system is to enable adding or modifying functionalities with minimum or no impact on the system.

In the VeriDevOps project, a roadmap and release plan has been defined for the integrated framework. The roadmap is organized according to a 2-phase work plan, where each phase includes the three main activities:

1. Integration activity, which belongs to WP1 and integrates the tools in the framework.
2. Toolset development activity, belonging to WP2-WP4 and which develops Security Requirements Generation, Reactive Protection at Runtime as well as for Prevention at Design and Development tools sets respectively.
3. Validation and evaluation activity, carried out in WP5 and which evaluates the tools, framework, and case studies.

The final release of the VeriDevOps Framework, due M36, will include all the final versions of every component in the framework. The evaluation of the architecture and implementation of the framework will be held according to the timeline shown in Figure 4.2, and the results will be presented in the final version of this deliverable.

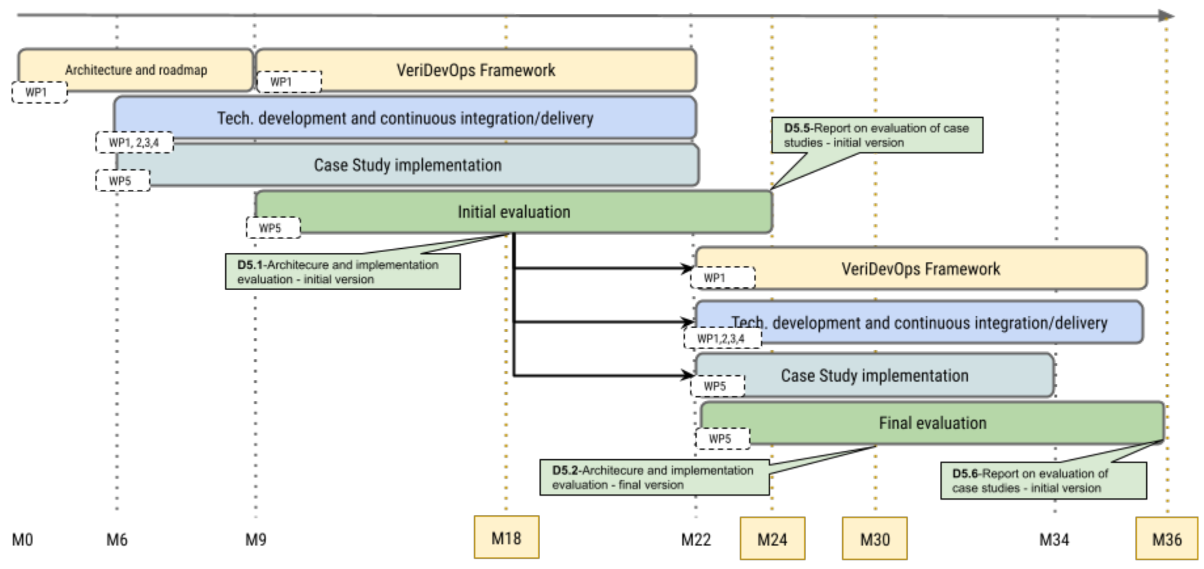
Architecture and roadmap
WP1
VeriDevOps Framework
WP1
Tech. development and continuous integration/delivery
WP1, 2,3,4
Case Study implementation
WP5
Initial evaluation
WP5
D5.5-Report on evaluation of case studies - initial version
D5.1-Architecure and implementation evaluation - initial version
VeriDevOps Framework
WP1
Tech. development and continuous integration/delivery
WP1,2,3,4
Case Study implementation
WP5
Final evaluation
WP5
D5.2-Architecure and implementation evaluation - final version
D5.6-Report on evaluation of case studies - initial version
M0
M6
M9
M18
M22
M24
M30
M34
M36

Figure 4.2 Architecture and Implementation Evaluation Timeline.

As mentioned, each of the activities is carried out in two phases: the initial version and the final version. According to this 2-phase roadmap, the integration of the VeriDevOps Framework will be carried out according to the following schedule:

- M9,- Architecture and roadmap (WP1)
- M18 - Architecture and implementation evaluation (WP5) - initial version
- M9-M25 - Tools sets (WP2-WP4) - initial versions
- M22 - VeriDevOps Framework (WP1) - initial version
- M24 - Case Study evaluation (WP5) - initial version
- M30-M36 - Toolsets (WP2-WP4) -final versions
- M30 - Architecture and implementation evaluation (WP5) - final version
- M36 - Case Study evaluation (WP5) - final version
- M36 - VeriDevOps Framework (WP1) - final version

5. Evaluation Plan

As shown in D1.1 and the integration roadmap, the execution of the baselining happens as industrial empirical studies that are structured according to the widely accepted Quality Improvement Paradigm (QIP)[4] and experimental software engineering practices[5]. QIP will be used as a framework for systematically managing KPI improvement by using experiments and industrial measurements. We modified the QIP process for our purpose. It consists of six steps that are repeated iteratively:

- **Characterize (WP1).** Understand the current industrial use case situation and establish a set of baseline case study scenarios that can be experimentally evaluated using qualitative and quantitative data collection - **MS1**.
- **Set tool and use case specific KPI goals (WP1).** Quantifiable KPI goals such as the ones shown in Table 9.1 are defined and given in terms of improvement with regard to all use case scenarios - **MS2** and **MS3**[6].
- **Choose the VeriDevOps process, methods and tools (WP2, WP3, WP4).** Based on the characterization and the goals, we choose the appropriate VeriDevOps supporting methods and tools to be used in the empirical study - **MS2 and MS3**.
- **Execute the empirical studies (WP5).** The studies are performed and the experimental results are collected for evaluation purposes. Controlled experiments, case studies or interview studies are used depending on the type of KPIs we are measuring - **MS3 and MS4**.
- **Analyze the data collected (WP5).** The results (current testing practices, possible challenges, and quantitative and qualitative findings) are studied and practical improvements are identified - **MS4**.
- **Package and Disseminate (WP5 and WP6).** The experiences are packaged to build the basis for continuous improvement - **MS4, MS5 and MS6**.
- **Features to be taken into account when choosing a test automation framework (WP2, WP3, WP4).** Knowing the important industry-reported features of adopted tools can help us to choose the right ones for a certain context.

During the industrial validation and demonstration phase in VeriDevOps, we will measure progress against a set of objectives evaluated (outlined in Table 5.1) against several key

[4] Basili, Victor R., and Gianluigi Caldiera. "Improve software quality by reusing knowledge and experience." Sloan management review 37 (1995): 55-5
[5] Wohlin, Claes, et al. Experimentation in software engineering. Springer Science & Business Media, 2012.
[6] These six steps of the case study implementation are mapped to work packages (WPx) as well as milestones (MSy).

performance indicators (KPIs) and the use case scenarios in which these are tackled (shown under Use Case Scenarios in Table 5.1). These will be used for validation using tool-specific and use-case-specific KPIs depending on the tools provided in different scenarios and used to show the industrial-specific improvement in terms of test efficiency and effectiveness.

Objectives	Key Performance Indicators	Use Case Scenarios[7]
O1. Automation in ensuring formally specified requirements in the DevOps context.	**KPI 1.1:** 70% reduction of mean time to enable protection and prevention means, compared with the manual approach. **KPI 1.2:** 50% reduction of mean time to resolution compared with manual approach.	FAG_S1, ABB_S2, ABB_S1
O2. Automated generation of formal specifications from security requirements.	**KPI 2.1:** F1-measure[8] of 0.7 in classifying the security requirements. **KPI 2.2:** Automated translation of 80% of security requirements into formal specifications.	FAG_S1, FAG_S3, ABB_S1
O3. Automated root cause analysis and counter-measures recommendation system at operations.	**KPI 3.1:** Decreasing by 50% mean time to detect anomalies. **KPI 3.2:** Decrease in the number of false positive alerts by 20% **KPI 3.3:** 100% of security requirements are traced at operations **KPI 3.4:** 20% increase in the number of detected and fixed security faults **KPI 3.5:** Decreasing by 50% average time to patch a vulnerability.	FAG_S2, ABB_S4, ABB_S6, ABB_S1
O4. Automated specification and verification of requirements.	**KPI 4.1:** 80% of security requirements tested automatically **KPI 4.2:** 80% of vulnerability and domain-specific coverage[9] for test cases **KPI 4.3:** 20% increase in the number of	FAG_S4, ABB_S5, ABB_S3, ABB_S1

[7] These scenarios will be available

[8] F1 is a measure of a test's accuracy in statistical analysis of binary classification.

[9] attempts to evaluate with respect to the vulnerability kinds it can detect. In other words, it assesses the percentage of all possible vulnerability kinds covered (e.g., buffer overflow, source code disclosure).

	detected and fixed security faults before operations.	
O5. Integration of a set of innovative realistic application case studies.	**KPI 5.1:** Integration of 5 industrially-common tool sets for protection (Ops) and prevention (Dev) over formal specifications of security requirements.	Applies to all use case scenarios

Table 5.1 Project Objectives and KPIs in relation to Use Cases

The baselines are measured and indicated during the industrial specification and demonstration phase and reported in D1.2.

5.1. Use Case Scenarios

In this section, the case study scenarios will be briefly described together with the actual context in which the scenarios will be implemented. In any case, the scenarios will be described from a use case high-level perspective.

5.2. Fagor: Manufacturing Machinery

5.2.1. Software Test and Automation (FAG_S1)

During this case study, the objective is to define cybersecurity-related tests targeting both frontend and backend applications and execute them as a stage of the CI/CD pipeline. This execution will be made with every commit to the development branch and daily with the latest stable release. The defined cybersecurity tests should be compliant with international standards and certifications, such as ISA/IEC 62443[10] in order to move forward a step on future certifications of the IoT platform.

5.2.2. Security Monitoring (FAG_S2)

Through this scenario, intrusions and anomalies in the cloud network need to be detected. In the cloud environment, an MQTT service is exposed through the Internet to receive telemetry data coming from the machines. No automatic monitoring is performed about the connections on the MQTT broker, and FAG doesn't know if there are clients without appropriate certificates trying to send malformed data or other kind of attacks to this service.

[10] https://www.isa.org/products/isa-62443-1-1-2007-security-for-industrial-automat

If a security issue is detected, its root cause should also be identified and a counter-measure(manual or automatic according to the threat impact) should be also recommended/applied.

5.2.3. Alignment of the Edge Device configuration to the NIST Framework (FAG_S3)

The objective in this Use Case scenario would be to implement a module, which will interpret the NIST framework requirements and check in the system if the requirements are implemented or not. It will try to implement the ones needed and raise notifications about the actions carried out.

5.2.4. Vulnerability verification and correction suggestions (FAG_S4)

FAG wants to deliver a complete product that stays secure over time. With that purpose, FAG is in need of having 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. This process will raise notifications about the actions carried out, pointing out priorities and vulnerability severity.

5.3.ABB: Automation Industry and Operation of Cranes

5.3.1. ABB MP Test Process Assessment (ABB_S1)

In this assessment of the ABB MP test process, the feasibility of how requirements are written from a test perspective will be evaluated, the levels of testing, selection of test cases and test steps, test coverage, and testing from a validation perspective.

5.3.2. From Requirements and Design to Test Specification (ABB_S2)

The vision is that the gap between the requirements, design, coding and testing can be closed. Several VeriDevOps methods and tools allow for designing a system on several hierarchical layers (e.g. functional and module levels). Artifacts are defined on a higher level and are linked to or be reused on lower levels. Different kinds of views on the system can be designed by using appropriate requirement types.

5.3.3. Test Design and Automation in Codesys with IEC 61131-3 (ABB_S3)

There are several areas of testing which can be potentially addressed in the context of VeriDevOps. The overall goal is to start validation early in the development process, to increase consistency and the quality of test cases, and to automate the generation of test cases and the testing itself.

5.3.4. Security Behaviour Analysis using Execution Traces (ABB_S4)

In this scenario, we are targeting the detection of 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. Furthermore, we will determine a list of potential countermeasures that need to be applied to enforce security at runtime.

5.3.5. Anomaly Identification and Handling of Safety and Security Requirements (ABB_S5)

By configuring or learning the typical behavior of a particular crane it should be possible to detect if the crane is in a hazardous state and by that avoid an accident. A similar method could also be used to detect a security breach or a cyber attack. The expectation is to develop VeriDevOps methods and devices to increase safety and security by monitoring network communication or devices' internal state.

5.3.6. Known Anomaly and Vulnerability Identification (ABB_S6)

Vulnerabilities identification should cover not only control software, but also software tools for development and maintenance, as well as hardware and device configuration. The expectation in VeriDevOps is to be able to monitor these known vulnerabilities with the aim of ensuring security in operation.

Here we outline the work needed to achieve this by focusing directly on one of the two use cases (and the use case scenarios) and the tool evaluations.

5.4. Use Cases Related KPIs – Initial Version

Given each use case scenario outlined in Table 5.1, here we show the use-case specific KPIs, their initial values as well as the targeted values (TBE stands for To be Estimated):

Table 5.2 Use case related KPIs

Scenario	KPIs Descriptions	Initial Value (e.g., Baseline if exists or can be estimated)	Targeted Value
ABB MP Test Process Assessment (ABB_S1)	a) How much of the safety standard ABB fulfills ba) How much of the security standard ABB fulfills	a) TBE b) unknown	a) 80-100% b) 80%
From Requirements and Design to Test Specification (ABB_S2)	a) time to write a set of requirements for a safety function (we have about 25-50 safety functions on a crane) that also fulfills the safety standard b) time to design and implement a safety function (we have about 25-50 safety functions on a crane) that also fulfills the safety standard c) time to create a test specification with a specified test coverage for a safety function	a) 40-60 man hours b) 40-80 man hours c) 40-80 man hours (test coverage excluded)	a) 30 hours b) 30-60 hours c) 30-60 hours (including test coverage defined)
Test Design and Automation in Codesys with IEC 61131-3 (ABB_S3)	a) time to include a new process/safety/security function in the test suite (incl. write test spec and script) b) coverage of functionality tested during a test execution suite (e.g. a regression test)	a) 2-3 of weeks b) TBE	a) 1 week b) a known coverage
Security Behaviour Analysis using Execution Traces (ABB_S4)	a) ability to detect known anomalies b) ability to detect new/unknown anomalies	a) TBE, manual effort b) TBE, manual effort	a) semi-automated tool support b) semi-automated tool support

Anomaly Identification and Handling of Safety and Security Requirements (ABB_S5)	a) ability to detect known anomalies b) ability to detect new/unknown anomalies	a) manual monitoring using watches in development tool and monitoring tools (e.g., wireshark) b) TBE	a) semi-automated tool support b) semi-automated tool support
Known Anomaly and Vulnerability Identification (ABB_S6)	a) ability to detect vulnerabilities in developed/delivered systems	a) TBE, manual effort	a) semi-automated tool support and a developed process for handling such cases
Software Test and Automation (FAG_S1)	a) How much time is saved detecting vulnerabilities. b) Time to detect newly published vulnerabilities c) Ability to detect vulnerabilities after a change/update in the system d) Time to detect vulnerabilities after a change/update in the system	TBE, manual effort	a) automated tool support
Security Monitoring (FAG_S2)	a) Ability to detect known anomalies b) Ability to detect new/unknown anomalies	TBE	TBE
Alignment of the Edge Device configuration to the NIST Framework (FAG_S3)	a) Automated translation of security requirements into formal specifications. b) Reduce implementation time for the security measure deployment	a) 0% b) Not measured	80%

Vulnerability verification and correction suggestions (FAG_S4)	a) Ability to detect vulnerabilities in production systems	It is not checked.	100%

5.5. VeriDevOps Evaluation of Tool Features

For all use case scenarios, the VeriDevOps tools will also be evaluated in terms of their applicability. We define here a set of tool features important for the adoption and deployment of VeriDevOps tools.

5.5.1. Definition of Features

Choosing the right tools is a non-trivial task for many practitioners[11]. Detecting the most important features to be considered while choosing a test automation framework is a very helpful aid for both industry and academia. As an example of how to instantiate these features in an industrial case study, Salari et al.[12] conducted a study on choosing a proper test automation framework for CODESYS Integrated Development Environment (IDE). During this study, they identified several generic and industry-reported essential features that should be considered when choosing a tool for industrial adoption based on three sources of information used (academic-related works, industrial input, and official documentation). We divided the discovered features into five different categories based on their focus, including Company Constraints, Maturity, Testing Functionalities, Framework Flexibility, and Usability.

- **Company Constraints:**
    - **Cost** indicates the cost model used (FREE, MIX, PAY)[13]
    - **Supported Platforms** specifies the platforms supported by the framework (Windows, Mac, Linux)

[11] Paivi Raulamo-Jurvanen, Mika Mantyla, and Vahid Garousi. Choosing the right test automation tool: a grey literature review of practitioner sources. In International Conference on Evaluation and Assessment in Software Engineering, pages 21–30, 2017.

[12] Mikael Ebrahimi Salari, Eduard Paul Enoiu, Wasif Afzal, Cristina Seceleanu. Choosing a Test Automation Framework for Programmable Logic Controllers in CODESYS Development Environment. Proceedings-2021 IEEE 15th International Conference on Software Testing, Verification and Validation Workshops, ICSTW 2022. 2022.

[13] Alessio Ferrari, Franco Mazzanti, Davide Basile, and Maurice Ter Beek. Systematic evaluation and usability analysis of formal methods tools for railway signaling system design. IEEE Transactions on Software Engineering, 2021.

- **Ease of Installation** indicates whether the framework installation requires installing other additional components or it covers all the installation requirements. (YES, NO, PARTIAL)
- **License Type** implies the type of license used for the test automation framework (e.g. Apache, MIT).

- **Maturity**:
  - **Industrial Usage** specifies the level of reported industrial usage in academic papers and reports (HIGH, MEDIUM, LOW)
  - **Stage of Development** indicates whether the framework is evolved through releasing different versions (MATURE), it is an academic or early version (PROTOTYPE) or it is new but has strong fundamental roots (PARTIAL).
  - **Input Functionalities:** Determines how the software artifacts are represented: Graphical User Interface (GUI), Textual Representation (TEXT), imported textual file (IMPORT).
  - **Supported Objects** feature was discovered by reviewing the tools documentation. This feature indicates the object types that are supported (UNIT, SOFTWARE, SYSTEM).
  - **Documentation and Report Generation** characterizes whether the automatically generated reports and documentation of a tool contain well-detailed technical details (COMPLETE) or only some summarized technical details are available (SUMMARIZED).
  - **Requirements Traceability** specifies if the framework can provide traceability between the related artifacts (YES, NO).
  - **Artifact Creation Time** relates to the time required to produce artifacts (QUICK, SLOW).
  - **Playback Record** indicates whether the tool can record sessions and playback these executions (YES, NO)[14].
  - **Import Support** specifies if the framework can import software artifacts using external files (Python, Java, NO)[15].
- **Framework Flexibility:**
  - **Backward Compatibility** indicates to which extent test scripts developed with previous versions of the CODESYS framework can be used in the current version (YES, NO, UNCERTAIN).

[14] Heidilyn Veloso Gamido and Marlon Viray Gamido. Comparative review of the features of automated software testing tools. International Journal of Electrical and Computer Engineering, 9(5):4473, 2019.
[15] Emil Borjesson and Robert Feldt. Automated system testing using visual gui testing tools: A comparative study in industry. In International Conference on Software Testing, Verification and Validation, pages 350– 359. IEEE, 2012.

- **Standard Input Format** specifies whether the language that is used for developing test cases is based on a standardized programming language or not (YES, NO).
- **Modularity of The Tool** specifies if the tool supports a wide range of different modules and add-ons that can be used to extend its functionality or not (YES, NO).
- **Teamwork Support** indicates whether the tool supports multi-user development and collaboration (YES, NO).
- **Framework Development Language** details the programming language that is used to develop the tool (e.g., Python, Java, C).

* **Usability:**
    - **Programming Skills** specify what level of programming skills is needed to work with the tool. Available options are advanced needed programming skills (ADVANCED), no programming skills required (NOT REQUIRED) or only required for advanced test scripts (PARTIAL).
    - **DevOps/ALM Integration Support** relates to the tool’s ability to support integration with DevOps or ALM environments (YES, NO).
    - **Continuous Integration (CI) Support** indicates if the tool supports CI frameworks (YES, NO).
    - **Report Format** indicates how the reports are represented in a tool (HTML, XML, CSV).
    - **Availability of Customer Support** evaluates the level of support and tutorials provided for the users of a certain tool (HIGH, MEDIUM, LOW).
    - **Graphical User Interface (GUI)** investigates the suitability of the designed graphical user interface of a framework. Available options are The GUI is designed properly and is powerful enough to cover almost all the available functionalities of a framework (YES); The provided GUI is user-friendly, but does not provide a graphical representation of all functionalities of a framework (PARTIAL); The GUI exists but it is limited in its functionality (LIMITED); the framework has no GUI (NO).
    - **Quality of Documentation** Specifies the tool’s level of the documentation and tutorial which is provided by the tool developers. Available options are: The tool is well-documented and a wide range of updated tutorials and tool specifications are easily accessible online (VERY GOOD); The tool is documented properly but the provided documentation is not easily accessible or it is only available offline (GOOD); The tool is not documented sufficiently or it cannot be accessed easily (INSUFFICIENT).

- **Maintenance Support** implies the level of maintenance support that is provided (EASY/HARD).

5.5.2. List of Features

Since our aim is to address the needs of industrial practitioners, we evaluated the validity of the discovered features by checking them with our VeriDevOps industrial partners. The list of the discovered and validated test automation framework features and non-validated ones as well as their category and source of extraction, are shown in Tables 5.3 and 5.4 respectively.

Industry-verified Features		
Category	**Feature**	**Extraction Source**
Company Constraints	Cost	Literature
	Supported Platforms	Literature
Maturity	Industrial Usage	Literature
	Stage of Development	Literature
Testing Functionalities	Documentation and Report Generation	Literature
	Playback Record	Literature
	Test Suite Support	Literature
	Test Suite Extension	Industry
Tool Flexibility	Teamwork Support	Literature
Usability	DevOps/ALM Integration Support	Literature
	Continuous Integration (CI) Support	Literature
	Script Language	Literature

	Availability of Customer Support	Literature
	Quality of Documentation	Literature
	Maintenance Support	Industry

Table 5.3: Extracted and Validated Framework Features

Other Features		
Category	**Feature**	**Extraction Source**
Company Constraints	Ease of Installation	Literature
	License Type	Tool Documentation
Testing Functionalities	Test Script Specification	Literature
	Supported Testable Objects	Tool Documentation
	Requirements Traceability	Literature
	Script Creation Time	Literature
	Import Support	Literature
Tool Flexibility	Backward Compatibility	Literature
	Standard Input Format	Literature
	Modularity of The Tool	Literature
	Framework Development Language	Literature
Usability	Programming skills	Literature
	Report Format	Literature

	Graphical User Interface (GUI)	Literature

Table 5.4: Other Extracted Framework Features

These features will be used in the final version of this deliverable for evaluating the overall applicability of the VeriDevOps tools and how they are applied to each use case.

6. Summary

This document presented an overview of the architecture of the VeriDevOps Framework along with a roadmap for the integration of each of the tools that are being developed in the project inside the overall architecture.

The deliverable presented an evaluation plan as well as KPIs to evaluate each of the components of the VeriDevOps framework and to compare the results to the ones expected by the case study owners as well as the overall project objectives. Since most of the tools are still under development process, the complete evaluation will be held in the final version of this deliverable, that is, “D5.2 Report on the architecture and implementation evaluation – final version”.