Instituto Universitário de Lisboa

Departamento de Ciências e Tecnologias da Informação

Large Scale Agile Software Development compliant to IEC 62443-4-1

Artefact Design and Tool support

Rafael Martins Soares

A Dissertation presented in partial fulfillment of the Requirements for the Degree of

Master in Computer Engineering

Supervisor

Doctor Maria Cabral Diogo Pinto Albuquerque, Assistant Professor

ISCTE-IUL

Co-Supervisor

Doctor Kristian Hendrik Beckers, Cyber security consultant

Siemens Corporate Technology

October 2019

Resumo

Nos últimos anos houve um aumento considerável no uso de metodologias ágeis. No entanto, a aplicação destas metodologias pode ser um desafio, em especial para sistemas de controle industrial que têm a obrigação de obedecer a requisitos operacionais rigorosos através de regulamentos e normas, e em particular no âmbito da cibersegurança. Este trabalho propõe um conceito para uma integração estruturada e sistemática de actividades de segurança num *pipeline* de DevOps, com o intuito de alcançar ambas as capacidades de desenvolvimento ágil seguro e engenharia de software ágil em conformidade com segurança. A base para este conceito é a integração da norma IEC 62443-4-1 (4-1), que descreve o desenvolvimento seguro de produtos em ambientes de controle industrial, com um especificação de *Continuous Integration/Continuous Delivery*. Para alcançar isto, foi feito um mapeamento de requisitos de segurança, de acordo com a descrição na norma 4-1, numa especificação simples de DevOps. Como resultado, todas as actividades da norma 4-1 foram analisadas e classificadas de acordo com a possibilidade de serem automatizadas através de suporte de ferramentas. Para avaliar o trabalho, foram realizadas entrevistas com profissionais especializados nas áreas de conformidade em segurança de TI's e engenharia de software ágil. Os resultados mostram evidências sobre a possibilidade de fornecer suporte de ferramentas para a automatização da norma IEC 62443-4-1 e para a especificação um *pipeline* de DevOps conforme com a norma 4-1.

Palavras-chave: Segurança em TI's, norma de segurança, segurança contínua, conformidade contínua.

Abstract

There has been a considerable increase in the use of agile methodologies over the last years. However, applying these methodologies can be challenging, particularly for industrial control systems that must obey to rigorous operational requirements through regulations and standards, and in particular cybersecurity requirements. The current work proposes a concept for a structured and systematic integration of security activities into a DevOps pipeline, with the ambition of pursuing the capability of both secure agile development and security compliant agile software engineering. The basis for this concept is the integration of the IEC 62443-4-1 (4-1) standard, which describes secure product development in industrial control systems, with a Continuous Integration/Continuous Delivery pipeline specification. To achieve this, the security requirements, as described in the 4-1 standard, were mapped into a simple DevOps pipeline specification. As a result, all of the 4-1 activities were analysed and classified according to the possibility of being automated through tool support. Interviews with expert practitioners, from the fields of security compliance and agile software engineering, were conducted to evaluate the present work. Results have shown evidence about the possibility of providing tool support for the IEC 62443-4-1 standard and to specify a DevOps pipeline compliant to the 4-1 standard.

Keywords: IT Security, security standard, continuous security, continuous compliance.

Acknowledgements

I would like to acknowledge my supervisors, Professor Maria Pinto Albuquerque and Dr. Kristian Beckers, for constantly guiding me and making this work possible. To all the people at Siemens CT for providing me a great work environment, especially to the working students, for all the ideas shared during the coffee breaks and for making this a more enjoyable experience. I also want to thank Fabiola Moyon, for being my unofficial supervisor, and Tiago Gasiba, for introducing me to Siemens CT and for all the advice he has given me during my time at the company.

A special thank you to my colleague André Narciso, for deciding to share this journey with me from the beginning. Without him, this experience wouldn't be the same.

Most importantly, I want to show my gratitude to my parents, for unconditionally supporting me, and for always motivating me to achieve greater goals in life.

Last but not least, I want to thank my older brother, Rúben, for always being there for me on crucial times, and for helping me keep both my feet on the ground.

Contents

List of Figures

List of Tables

Abbreviations

SDLC	**S**oftware **D**evelopment **L**ife **C**icle (see page 1)
CI/CD	**C**ontinuous **I**ntegration/**C**ontinuous **D**elivery (see page 2)
ICS	**I**ndustrial **C**ontrol **S**ystem (see page 3)
4-1	IEC 62443-**4-1** Standard (see page 3)
SAFe	**S**caled **A**gile **F**ramework (see page 3)
BPMN	**B**usiness **P**rocess **M**odel **N**otation (see page 4)
IAST	**I**nteractive **A**pplication **S**ecurity **T**esting (see page 14)
IACS	**I**ndustrial **A**utomation and **C**ontrol **S**ystem (see page 16)
SDL	**S**ecure **D**evelopment **L**ife-cycle (see page 16)
SL-C	**S**ecurity **L**evel-**C**apability (see page 38)
4-2	IEC 62443-**4-2** standard (see page 38)

Chapter 1

Introduction

This Chapter introduces the scope of the work by presenting an overview and background of the topic, followed by the motivation and the proposed objectives. An outline of the following chapters of the document is also presented in this Chapter.

1.1 Overview

Over the last years there has been a considerable increase in the use of agile methods. Agile methodologies promote continuous iterations in which requirements and solutions evolve throughout the software development life cycle (SDLC) through customer collaboration (Beck et al., 2001; Highsmith & Cockburn, 2001). Despite the fact that agile methods were originally seen as more appropriate to small and co-located teams, according to (VersionOne, 2011), a large-scale industry survey recorded that 80% of the studied organisations were already following an agile approach in 2011.

DevOps is a combination of cultural philosophies and practices that aim at breaking the barrier between software development (Dev) and information technologies operations (Ops) teams (Atlassian, 2016). It relies on Agile and Lean[1] techniques that require a comprehensive collaboration among stakeholders and has the intention to reduce the time between the commit of a change and its placement into production.

An essential part of DevOps is the concept of a Continuous Integration / Continuous Delivery (CI/CD) pipeline . This pipeline is a systematic use of aligned processes and tools that enable agile teams to release software by the push of a "button". The button triggers a set of automated checks and tests for the software that ensure its quality (Semaphore, 2019).

For this reason, there has been an increase in the adoption of DevOps techniques among enterprises, aiming for a faster time to market with the help of automation.

Applying DevOps in an industrial environment can be challenging, specially when it comes to industry operations and control systems. Not only these systems must obey to rigorous operational requirements, like high availability and real-time performance, they also must comply with regulations and standards for safety and security (Gartner, 2017; Sonatype, 2019). Enhancements to existing practices and innovative methods and concepts are fundamental to safely apply DevOps in an Industry environment. An example of such innovative methods and concepts are the creation of new architecture concepts for more robust, side-effect free deployment of products.

1.2 Motivation

Agile software engineering along with Continuous software engineering methodologies emerged with a closer cooperation with the customer, bringing numerous

[1]The core idea behind Lean is to maximize customer value while minimizing waste, i.e., create move value for customers with fewer resources (LEI, 2009)

benefits, such as faster software development and more feedback, allowing continuous innovation. Such methodologies still remain as a paradigm for software engineering in numerous domains (Lieberman, Paternò, Klann, & Wulf, 2006; Wang, Conboy, & Cawley, 2012; Fitzgerald & Stol, 2014).

However, software engineering for domains that have a high demand for security such as industrial control systems (ICS) , have yet several challenges to overcome before agile methodologies can be largely applied. Specially software for security critical systems that plays a vital role in supporting modern society and that has to be often engineered to be compliant to security standards. This software needs various security analysis and risk management activities, making secure development harder and less affordable (Alcaraz & Zeadally, 2015).

The ICS domain is regulated by security standard family IEC-62443 (ISA, 2017) and the secure software engineering process by IEC-62443-4-1 (4-1) in terms of cybersecurity.

Nowadays software engineering methods are lacking the ability to provide compliance between these standards and large scale agile methodologies, such as Scaled Agile Framework (SAFe) (Scaled Agile, 2017).

Additionally, a challenge that is currently emerging in the software engineering field is the recurring trade-off between implementing security into software engineering projects and competition with the speed of digital business, due to the ability to automate tasks and processes.

1.3 Objectives

The present document proposes a concept for a structured and systematic integration of security activities (according to the IEC 62443-4-1 standard) into a DevOps Life cycle in order to tackle the current security and compliance challenges without taking a step back in respect to time to market / release cycle. The basis for the concept is the integration of the current 4-1 standard for secure product

development in Industrial Control systems (ICS) and its integration into a CI/CD pipeline specification.

Since DevOps pipelines are tool chains for software development that enable CI/CD of software products, the specification shall serve as a basis for an assessment of pipelines to check if they have the capabilities to be 4-1 ready and to provide a detailed list of missing tool capabilities in order to achieve this readiness. In addition, it shall enable large-scale agile development for secure software according to the 4-1 standard in an industry environment.

The current project for integrating the 4-1 standard security requirements into a product development pipeline was developed at SIEMENS CT, which is an industrial environment that largely relies on automation and there is the need to strictly follow security best practices. Since the 4-1 is the current standard for secure product development in an ICS environment, SIEMENS CT provided a visual representation of the standard in the form of Business Process Model Notation (BPMN) models developed by (Moyon et al., 2018) to describe the workflow of secure product development. The BPMN models break down the security requirements into activities, which will serve as a basis for the Continuous Integration / Continuous Delivery (CI/CD) pipeline specification. Furthermore, the specification is intended to contain the flow of information regarding secure product development and discuss which activities can be automated with the help of tools and which cannot.

By the end of the present work it shall be possible to answer the following research questions:

- **RQ1 - Is it possible to create a DevOps Pipeline that follows a systematic way to security compliance with IEC 62443-4-1 standard?**
  - **RQ1-a - How much tool support can we provide for 4-1 compliance?**

1.4 Outline

The present document is divided into six chapters, which contain:

- The following chapter, **Chapter 2**, introduces the fundamental concepts for the understanding the present work, as all as an analysis of the previous work realised under the current work research area. The main research domains included are DevOps and pipeline specifications, DevSecOps and security compliant product development.
- **Chapter 3** describes the proposed concept for combining the 4-1 standard with a DevOps pipeline.
- **Chapter 4** starts by describing a methodology to achieve the integration of the 4-1 standard and the pipeline, in particular the definition of templates for automation tool support. Later, it is described all the steps taken towards the application of methodology, as well as an observation and an analysis of the obtained results.
- **Chapter 5** presents the evaluation of the obtained results through surveys and interviews with security practitioners and professionals.
- Finally, **Chapter 6** draws the conclusions of the present work, the difficulties that emerged during its execution and recommendations for future work.

Chapter 2

Literature review

This Chapter presents a description of fundamental concepts to understand the present work, as well as a detailed analysis of work relevant to the area of study. The first sections focus in the field of software development, product development life cycle and agile methodologies, in particular DevOps. Subsequent sections focus on security compliant product development and security standards, especially the IEC 62443-4-1 standard. Furthermore, a key contribution of the IEC 62443-4-1 standard is described, as well as the reasoning for its adoption into the specification of a deployment pipeline.

2.1 Software development life cycle

A Software Development Life Cycle (SDLC) corresponds to a systematic process for creating software. Its goals are to ensure that the software is built with quality, correctness and that it meets the customer expectations (Techopedia, 2011).

There are different methodologies or models that will differ from one another, however they all share the same purpose of helping teams to deliver high quality software as fast and cost efficient as possible. Some of the most popular models are the Waterfall, the V-model, the Spiral and the Agile model, which is the one

with higher focus in this project and described in the first Section of the previous Chapter.

SDLC contains different phases and each of them has it own activities and deliverables that are mandatory to start the next phase. Figure 2.1 shows an overview of the different SDLC phases and its interactions, regardless of the chosen methodology.

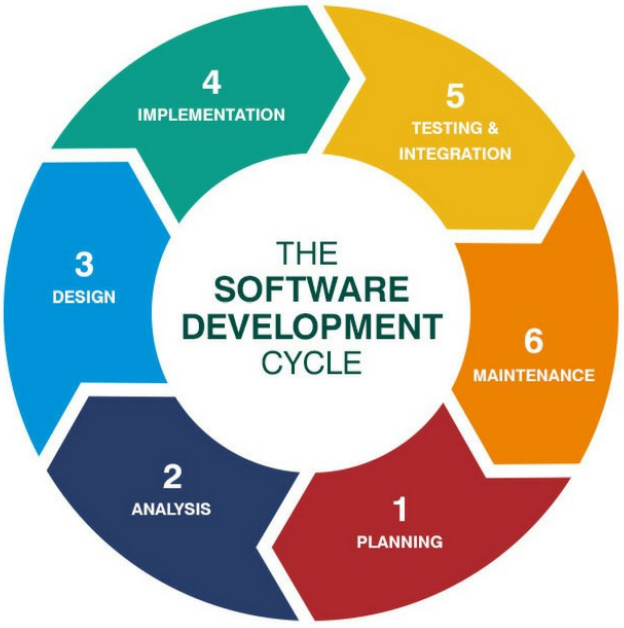


FIGURE 2.1: Software development life cycle phases, adopted from (Synotive, 2017)

It is possible to have different SDLC phases, however this is an overview and other phases may be similar regarding processes, deliverables and interactions between them. The different phases of the SDLC as represented in the Figure 2.1 are the following:

1. **Planning** and **Analysis** are the first phases of the software development life cycle and are crucial for the development of a project. This is the phase which gives a clearer idea of the scope of the project where anticipated issues and opportunities are assessed. Furthermore, schedules, cost estimations and requirements need to be detailed precisely so that the organisation can set a timeline to finalise the project. If following a waterfall methodology, it

is usually a document that lists all these requirements. In case of an agile methodology, its possible that a backlog of tasks to be performed is produced.

2. In the **Design** phase, the system and software design are prepared and documented according to the requirements gathered in the previous phase. This will also help in specifying requirements for the hardware and software, as well as defining overall system architecture. These specifications serve as an input for the following phase and testers also come up with a strategy where it is decided what, why and how to test.

3. **Implementation** phase, also known as coding or software development phase. Here's where developers need to work according to predefined coding guidelines in order to produce the code that will build the entire system. This code is then used as an input for the following phase.

4. **Testing & Integration** phase starts once the code from the previous phase is developed. The code is then tested against the requirements to guarantee that the product meets the customer's needs. Similarly explained in Figure 2.3 in the previous Section, during this process the code must pass different types of functional and non-functional testing. This will result in functional software that is ready to be deployed for customer usage.

5. **Maintenance** phase begins once the customers starts using the developed system. The main objective of this phase is to guarantee that the system continues to function according to the mentioned specifications in the early phases and that the needs continue to be satisfied. In this phase this phase it is possible that the customer finds bugs that need to be fixed, or asks for new features into the existing software. For this reason, the SDLC is an ongoing process, as seen in Figure 2.1.

It is important to note that specific SDLC models may not just perform a single cycle, as there may be several iterations. An example of this is the incremental model, in which the process of development is broken into multiple iterations that

pass through the different SDLC phases. Each subsequent iteration adds up to the previous one until all the designed functionalities have been implemented.

2.2 Deployment pipeline

A deployment pipeline is a manifestation of every process that is needed to transfer a concept into the hands of the user, which allows the visualisation of progress as it moves from stake to stake, until it finally reaches release (Humble & Farley, 2010). Figure 2.2 is a simple illustration that shows the entire process of a deployment pipeline, where it is possible to see the elapsed time in each of the stages.

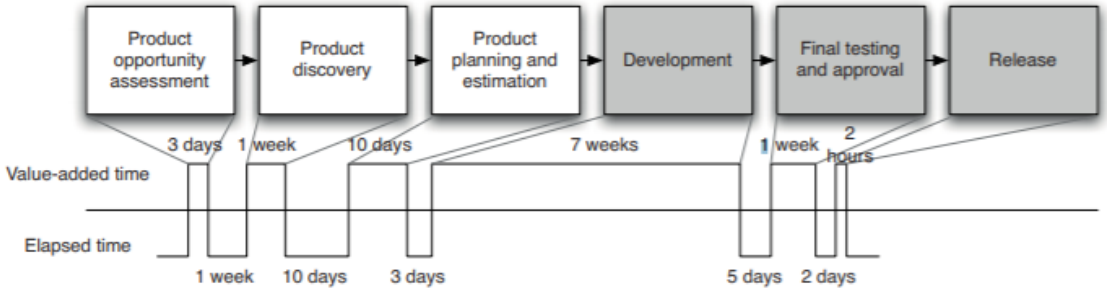


FIGURE 2.2: A simple value stream map for a product, adopted from (Humble & Farley, 2010)

It is important to notice in this picture of a pipeline that there are dependencies and waiting times between stages, making the "Elapsed time" higher than "Value-added time"[1] (e.g., the time it takes to deploy an application to production-like environment).

To get a better understanding of the dependencies and the flow of the pipeline, Figure 2.3 corresponds to the "drill down" of the grey shaded stages from Figure 2.2. It is possible to see that every new build and/or change has to go through a series of tests and has to pass them in order to be released. This way it is possible to receive early feedback to remove unfit builds and work on the root cause of their failure.

[1]Value-added time means the time in which actual work is being done.

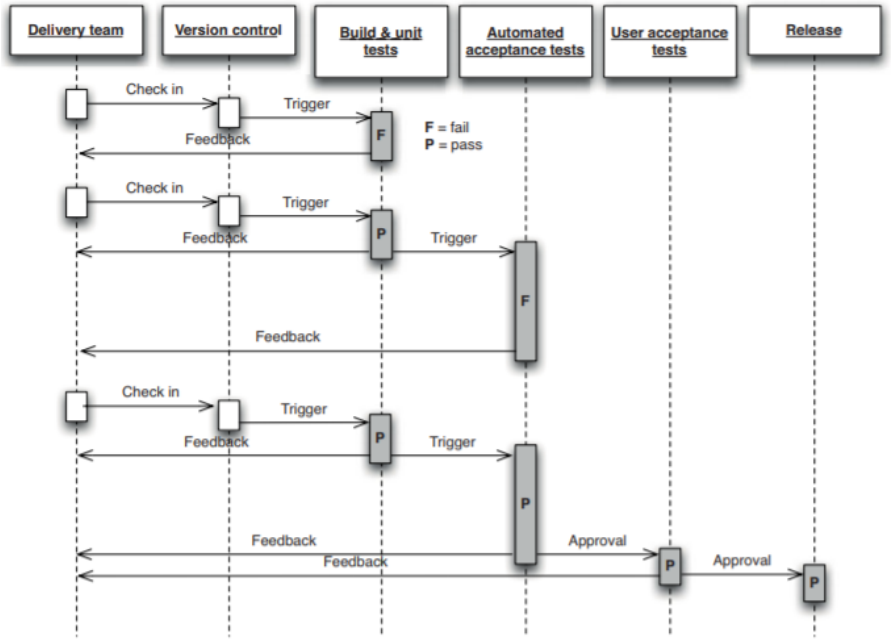


FIGURE 2.3: Changes moving through the deployment pipeline, adopted from (Humble & Farley, 2010)

For the most part, deployment pipelines share a structure that is similar to the one illustrated in Figure 2.2 and Figure 2.3, that contain a set of subsequent stages, from "Product opportunity assessment" to Release".

2.3 DevOps

A literature survey and interviews with practitioners was conducted at (Lwakatare, Kuvaja, & Oivo, 2015) in order to understand the phenomenon behind DevOps. The study identified four main elements in DevOps, which are collaboration, automation, measurement and monitoring. The noticed outcomes were the shared responsibility among teams, continuous deployments and operational data to measure performance of development.

(DORA, 2018) conducted scientific studies over the last years with over 30.000 survey responses, counted with almost 1.900 professionals from all around the world that participated in the 2018's State of DevOps. The report aims to understand the practices that lead to higher performance in software delivery, and

to help teams benchmark them-selves into four categories, low, medium, high and elite performers. For this purpose, they took into account four main metrics to measure the performance, which were deployment frequency, change failure rate, time to restore a service and lead time for changes (i.e. time elapsed from code commit to production). Some of the key findings include the use of open source software from high performers, increase in software delivery performance with the use of cloud infrastructure and the fact that outsourcing by function is rarely adopted by the highest performers. Some of the key technical practices that drive to high performance include monitoring and observability, continuous testing and integrating security earlier in the software development process.

The company that created the tool Puppet has been conducting studies over the last seven years, with over 30.000 technical professionals participating. (Puppet, 2018) has recently published a "state of DevOps report" in order to identify DevOps evolution, and key practices to help teams achieve success and progress in their journey. Some key findings stated in the report include the view on DevOps progress and impact regarding the C-suites and the teams that they manage. Another one that is highly mentioned is that automation of security policy configurations is critical to reach the highest levels of DevOps evolution, which requires breaking down the barriers between operations and security teams.

To this extent, over the last years deployment pipelines and workflows have been suffering changes in order to adapt to current needs and trends, including Continuous Integration and Continuous Delivery. This meant an increase in the demand of security in IT over time, with multiple attempts to reinforce and inject security into the CI/CD pipelines and workflows, such as addition of security checks and controls. (Bird, 2016) represents a good example of this, as illustrated in Figure 2.4. Figure 2.4 shows a clear mapping of some IT security practices and activities over all the different stages, from Pre-Commit to Production, providing a security guideline and assuring its application.

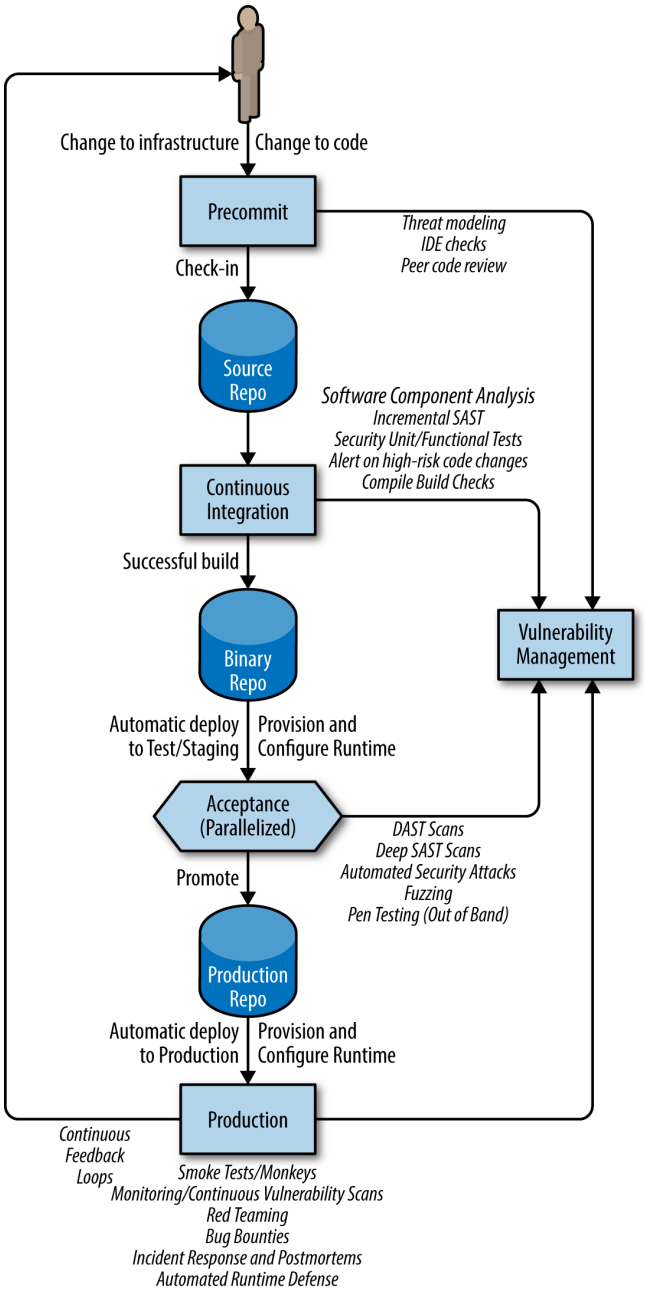


FIGURE 2.4: Security checks and controls in engineering workflows , adopted from (Bird, 2016)

2.4 DevSecOps

The term "DevSecOps" has emerged as organisations, that started to apply DevOps techniques, were concerned about the security aspects of software development. The term is also mentioned as "SecDevOps", "SecOps" and "RuggedOps" and it refers to the incorporation of security practices in a DevOps environment

through the collaboration between development, operation and security teams (Mohan & Othmane, 2016).

In order to have a better understanding on practitioners' perceptions and practices to integrate security in DevOps, (Rahman & Williams, 2016), analysed several articles and surveyed representatives of nine organisations. Some positive observations included leveraging on DevOps automation practices for monitoring, testing and deploying, and active collaborations between security, development and operation teams. It was also noted a correlation between the use of some non-automated security activities with security awareness among established DevOps organisations. On the other hand, it was observed that unrestricted collaboration can lead to inappropriate access to system resources.

(Gartner, 2017) presented key challenges and recommendations to aid the integration of security into DevOps. Challenges identified include the fact that product development teams, that deliver new IT-enabled capabilities, satisfy more the customers, rather than information security or IT operations. It was also indicated that organisations producing new applications and services using DevOps have the same responsibility to produce secure and compliant code as required by any other application, as well as, information security must adapt to development processes and tools. As for recommendations, it targeted security and risk management, which should focus on ensuring application and data security. The recommendations included the integration of security and compliance testing into the DevSecOps, and the scan for known vulnerabilities and misconfigurations in all open-source and third party components. Another recommendation was to be open to new types of tools and approaches to minimise friction for developers, such as interactive application security testing (IAST) to replace traditional static and dynamic testing.

(Gartner, 2017) also recommends that security doesn't stop in development, and that in order to achieve DevSecOps, the entire life cycle needs to be secured, from development to operations, as illustrated in Figure 2.5

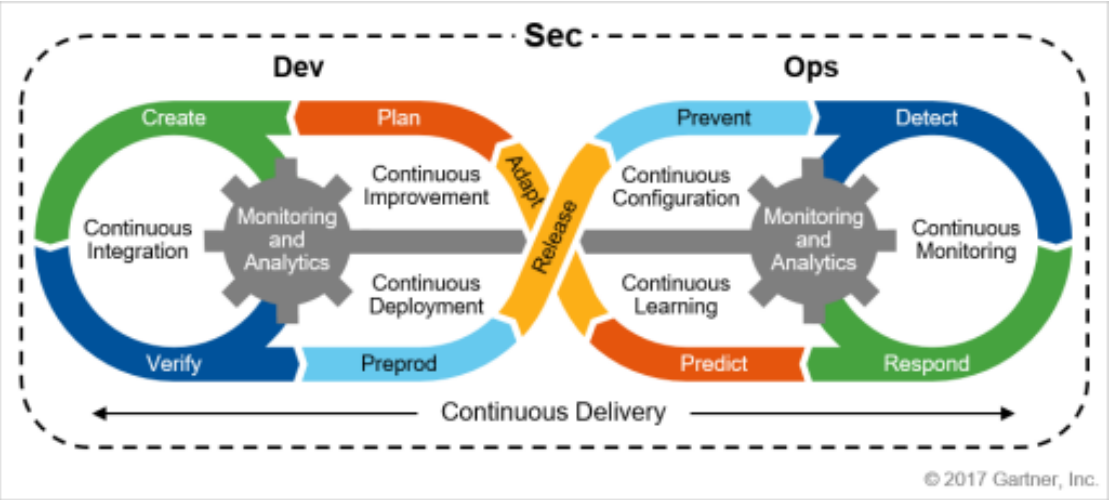


FIGURE 2.5: DevSecOps: Secure Development as a Continuous Improvement Process , adopted from (Gartner, 2017)

More recently, (Sonatype, 2019) conducted a survey with over 5.000 participants, whose responses were divided into individuals that work in a mature DevOps organisation and those who do not. Key findings include that DevOps teams are more likely to success, if they have a fully integrated and automated security practices across the DevOps pipeline, with increased feedback loops that enable security issues to be identified directly from tools. For top challenges related with security processes it was identified by developers that they receive security information late in the process, which leads them to rework and slowing down development. If security is not enforced in the processes, security is often integrated via impractical workarounds. The workarounds can cause various quality problems with the software.

2.5 Security compliant product development

One of the core characteristics of regulated environments is the need to comply with formal standards, regulations and guidance. For this reason, it was deemed important to investigate previous works and studies on security compliant product development.

2.5.1 Security standards

Standards were founded as a global agreement about the best approach to do something and there are currently standards that cover a whole range of sectors, from food safety management to secure medical packaging. Addressing standards can give a consumer the confidence that a product is safe, reliable and has good quality (ISO, 2017).

(Nair, de la Vara, Sabetzadeh, & Falessi, 2015) have conducted a study in order to get an understanding on how practitioners deal with safety evidence management for critical systems with safety standards. To do so, a survey was conducted among 52 practitioners from 15 different countries. According to the answers obtained from the survey, it was indicated that the most frequent safety evidence types are verification and validation test artefacts, requirements and design specifications. It was also noticed that evidence completeness checking and impact analysis are predominantly performed manually with the aid of expert judgement. It was concluded by the study that industry will certainly benefit from more tool support for collecting and manipulating safety evidence.

2.5.2 IEC 62443-4-1 standard

The 4-1 standard belong to the ISA99/IEC-62443 series of standards that address security for industrial automation and control systems (IACS) (ISA, 2018b), there standards were built according to several sources regarding good practices in software engineering and secure development life-cycle (SDL), with a special focus on (ISCI, 2018). Figure 2.6 illustrates the current status and the scope of the IEC-62443 series of standards.

As illustrated in the Figure 2.6, the IEC 62443 series of standards is divided into four groups, which cover the different roles and responsibilities of stakeholders of an IACS cyber security program (IEC, 2014). The first group, **General**, contains elements that address topics that are common across the whole series, such as terminologies, concepts, abbreviations and use cases. **Policies & Procedures**

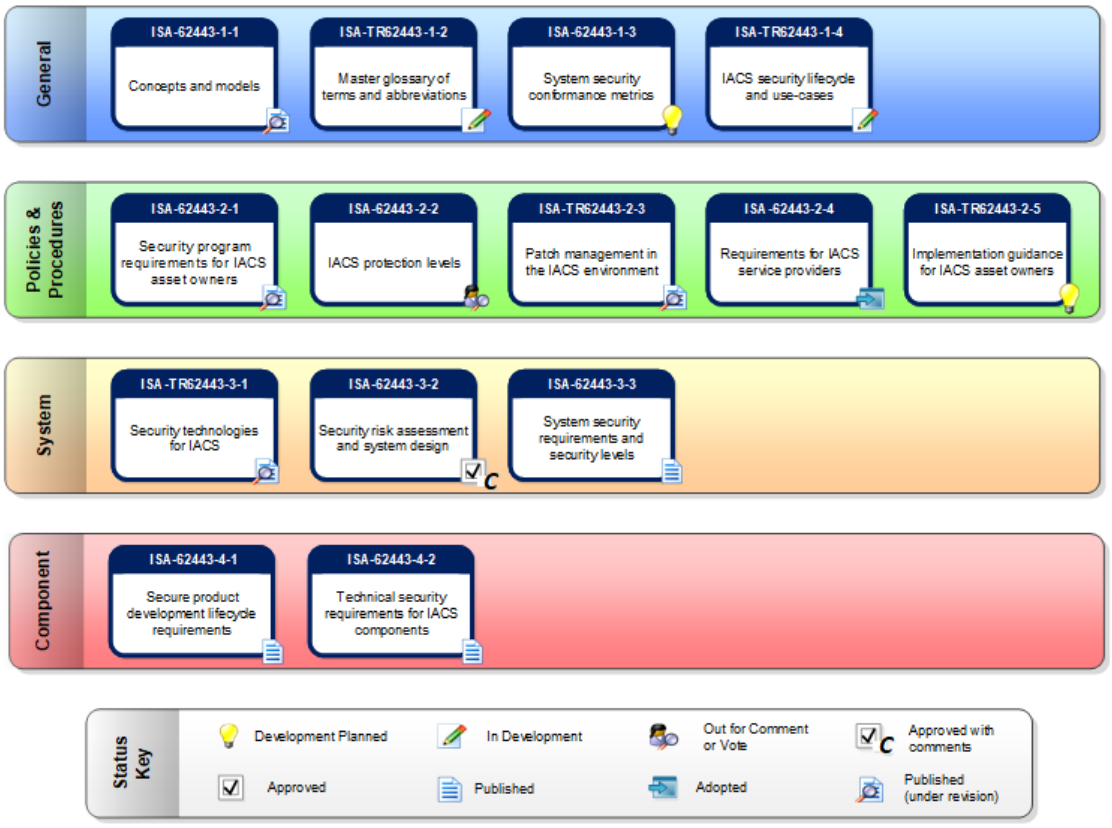


FIGURE 2.6: Scope of the ISA99/IEC 62443 series of standards , adopted from (ISA, 2018a)

group is focused on management and the asset owner, includes standards whose target are to describe policies and procedures required for consistent operation of a cyber security management system. **System** group is targeted for the system integrator, includes requirements for security of a system in an IACS environment. Finally, **Component** group describes the security requirement of a component in an IACS environment, focused for the product supplier.

The 4-1 standard, which belongs in the **Component** group, provides descriptions of requirements and guidance to enable secure product development life-cycle for products intended for use in an IACS environment (IEC, 2018). Hence, the targeted audience for the standard are product development organisations and developers of automation and control products that need to care about security.

The requirements from the 4-1 standard are divided in eight practices, which are ordered as the following:

1. **Security management** (SM) - This practice contains descriptions of processes that ensure that security-related activities are adequately planed, documented and executed through the entire product's life cycle. Once these activities are planned, they can be executed and documented through each of the secure development life cycle's practices.

2. **Specification of security requirements** (SR) - Describes the process to document the security capabilities that are required for a product, according to its security context. These security capabilities concern both physical and cyber security, such as authentication, authorisation, encryption, auditing and network security.

3. **Secure design** (SD) - Contains descriptions of processes to ensure that the product is secure by design. These processes should apply from the overall architecture to the design of individual components. This practice states that all of the product's physical and logical interfaces shall be identified and characterised.

4. **Secure implementation** (SI) - This practice contains processes to ensure that the product is implemented securely. These include requirements that apply to the product supplier for hardware and software development.

5. **Security verification and validation testing** (SVV) - This practice describes the processes that ensure that security testing is performed and documented. Security testing ensures that the product achieves its security requirements and that it is protected according to its security context.

6. **Management of security-related issues** (DM) - Contains descriptions of processes to handle product's security related issues that affect the product according to its security context. A process shall be employed to receive and track to closure security related issues reported by internal and external

sources, such as security testers, third-party supplies, developers and product users.

7. **Security update management** (SUM) - Describes the processes that ensure that security updates related to the product are tested for regressions and are delivered to users in a timely manner. The product supplier shall verify that the developed security updates solve intended vulnerabilities and that the update is not contradicting other operational, safety or legal constrains.

8. **Security guidelines** (SG) - Contains descriptions of processes that ensures that the user is provided documentation in order to integrate, configure and maintain the product.

2.5.3 Foundation of 4-1 in process models

In collaboration with SIEMENS CT, (Moyon et al., 2018), focused on removing the ambiguity from the 4-1 standard and described it in a visual representation in the form of Business Process Model Notation (BPMN) models. Figure 2.7 illustrates an BPMN model that presents an overview of the 4-1 standard by representing all of its previously mentioned practices and its interactions between each other over time.

As shown in the Figure 2.7, the process models are divided into eight sections, with each of the sections corresponding to a practice from the 4-1 standard. Each of the sections contains a detailed sequence of events, tasks and generated artefacts that are necessary to be 4-1 compliant. Additionally, each of these items includes a brief description.

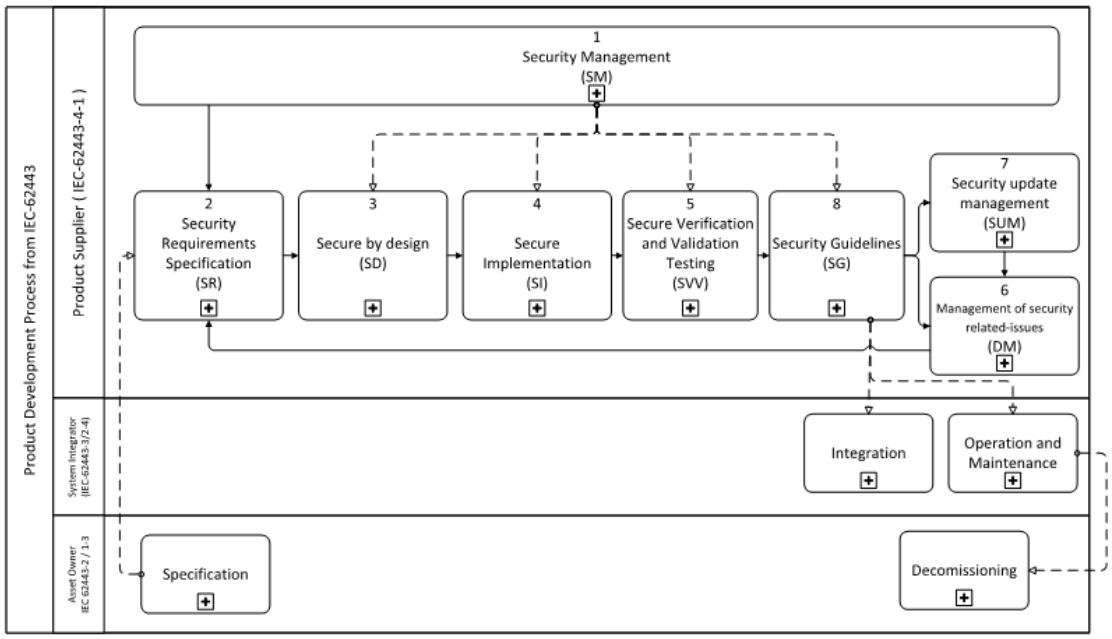


FIGURE 2.7: IEC 62443-4-1 Practice Overview as BPMN Model, adopted from (Moyon et al., 2018)

2.5.4 4-1 standard compliant solutions

The stated objective of this work is to create a 4-1 security standard compliant specification and for this reason it is important to investigate previous works that have found a solution that integrates this specific standard.

(Moyon et al., 2018) suggested a new approach to achieve continuous and secure development in security domains, specifically in industrial and automation control systems. To do so, (Moyon et al., 2018) proposed a model based approach, which consisted on merging visual representations of security norms and process models, in particular the SAFe framework and the IEC 62443-4-1 standard, resulting in a BPMN model. The use of a graphical modelling language was capable of removing the ambiguity generated by natural language, allowing for an easier review and more focused discussions among experts. It was concluded that more comprehensive models and validation with experts were required, however, initial results received a positive feedback from practitioners, judging that the models were suitable for what they were intended.

Furthermore, (Dännart, Constante, & Beckers, 2019) have created an assessment model specifically tailored for the 4-1 standard and the SAFe framework. The proposed model assessment meant to detail the development process activities and artefacts, and merged them into a compliance matrix in order to evaluate the compliance distance to the 4-1 standard. Goals and metrics were introduced by the creation of "requirement cards", which contained the expected input and output artefacts to enable precise assessment and easily identify which activities and artefacts have to be introduced or improved in order to comply with the 4-1 standard. Thus, providing a foundation for business-driven security compliance management by enabling a estimation of security compliance costs.

Chapter 3

Concept for combining the 4-1 standard and pipelines

This Chapter details the idea that leads towards the integration of the 4-1 standard with a DevOps pipeline as proposed in the first Chapter. Section 3.1 starts by giving a first impression of the integration of the 4-1 standard with a DevOps pipeline with an illustration of an initial high-level specification. Section 3.2 provides an insight on how to extend the previous process models into pipeline specifications, which includes two illustrations, one for an overview of the specification and another one for a specific practice specification. Finally, Section 3.3 describes how the specification shall provide proof of security compliance along the pipeline.

3.1 Integration of the 4-1 standard into the pipeline

Identically to (Moyon et al., 2018), the integration of the 4-1 standard into the pipeline consists on creating a visual representation of the specification, i.e., merging the 4-1 standard into a DevOps pipeline. To do so, it was fundamental to extensively analyse the purpose, as well as identify all the events and tasks that are attached to each of the 4-1 standard practices. Furthermore, the works described in the Chapter 2 play an important role in obtaining a better understanding

of the common structure of a DevOps pipeline, as well as the stages and the flow of processes.

In fact, the first step for integration of the 4-1 standard into the pipeline consisted in identifying the characteristics in common in both DevOps pipeline and the 4-1 standard. Afterwards, it was possible to merge each of the 4-1 standard practices into the corresponding stage of the DevOps pipeline. Therefore, allowing the design of an initial high-level 4-1 compliant pipeline specification, as presented in Figure A.1 (see Appendix A), thus making it possible to turn the process models into pipeline specifications.

3.2 Evolving the process models into pipeline specifications

With the current DevOps challenges in mind that were stated in the previous Chapter, this section describes how this approach emerges to overcome such challenges by easing the developers' work and adapting security by integrating security tools and practices directly into the pipeline, as well as covering the entire security life-cycle with a combination of automated and manual tasks.

In order to achieve this, it is essential to define and create templates prior to the creation of all the practice specifications. The templates detail the level of automation that is possible to achieve for each of the requirements from all the 4-1 practices. Each requirement contains an activity, which can be an event or a task and is detailed with the necessary input, output, description and extent of possible tool support, as well as the identification of the tool. Each of the templates has a specified output that functions as a quality gate, which can assure that security activities were executed or have the expected results. Generally speaking, a quality gate is a milestone that is located between two dependent phases in a pipeline and contains a list of activities or checklists verifying if the

requirements from one phase were fulfilled in order to decide if the pipeline should continue to the following phase.

3.2.1 Specification overview

With the given information described in the previous section, this section shows an overview of the specification. The Figure 3.1 illustrates how it is possible to integrate each of the practices in the 4-1 standard into the DevOps pipeline by creating new activities that allow 4-1 compliance. Each of the activities in the specification includes the identification of the practice, the necessary input, its description and the resulting output, according to the template, as stated in the previous section. Each of the added activities into the pipeline also includes a list of existing tools that help fulfil and automate the activity, according to the previously defined levels of automation. Thus, aiding and enabling secure software development by including security from the first to the last stage of the DevOps pipeline.

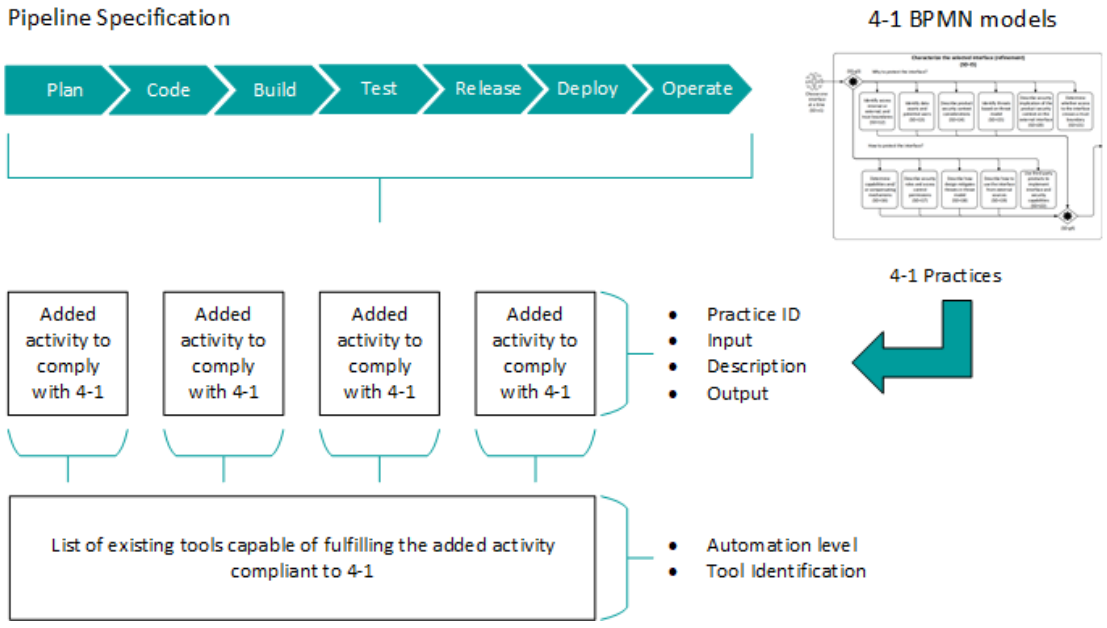
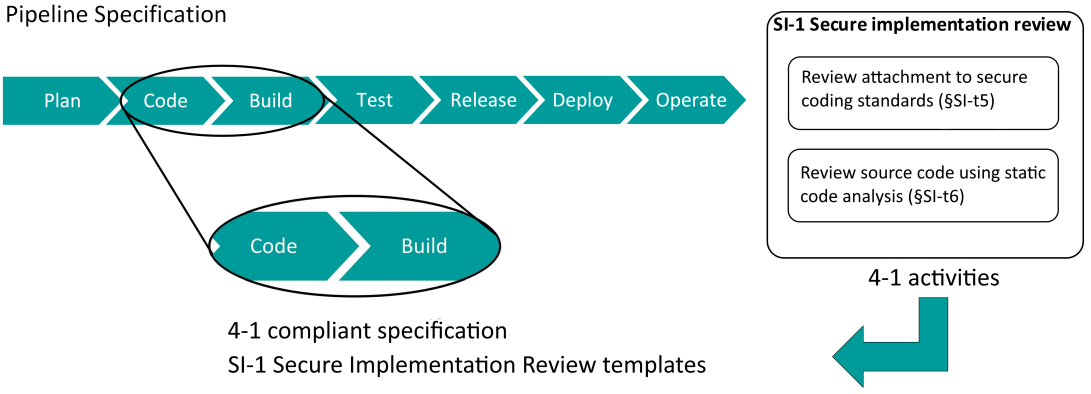


FIGURE 3.1: Evolving Process Models into Pipeline Specifications

3.2.2 Practice specification

Once it is identified into which stages of the pipeline the 4-1 practices belong, as referred in section 3.1, it is possible to create practice pipeline specifications. Figure 3.2 shows an excerpt, which corresponds to a drill down of the specification overview and may serve as an example of how 4-1 practices can be merged into a specific stage of a DevOps pipeline. In this specific case, it is possible to observe that tasks from the **Secure Implementation Review** requirement from the **Secure Implementation** practice were introduced into the **Code and Build** stage(s) of the DevOps pipeline. The illustration doesn't show all the necessary tasks for this requirement, as it solely serves the purpose of demonstrating how to merge the 4-1 practices along with the automation tool templates into the DevOps pipeline.



ID	SI-t5
Input	Secure coding standards (SI-a1)
Task	Review attachment to secure coding standards
Output	Validation from reviewer
Automation	Partial
Tool	OpenSCAP

ID	SI-t6
Input	Code
Task	Review source code using static code analysis
Output	Code review results
Automation	Complete
Tool	Findbugs; 360 Fireline; ESLint; Phan

FIGURE 3.2: Excerpt of the specification of the integration of the 4-1 standard with the pipeline

3.3 Ensuring proof of security life-cycle compliance

The integration of the 4-1 security activities into the product development life cycle is not complete without actual proof of compliance each time that a product is developed according to this pipeline specification. As mentioned earlier in the Section 3.2 of this Chapter, the output of each of the 4-1 activities has the function of a quality gate (e.g., documentation for security testing, vulnerability scoring system artefact). For this reason, for each activity demanded by the 4-1 standard, it is needed to have the capability of logging that the activity has been accomplished. Therefore, each tool should have the capability to store the results of the 4-1 activities in a central version control system to guarantee proof of compliance. Figure 3.3 serves the purpose of representing this idea.

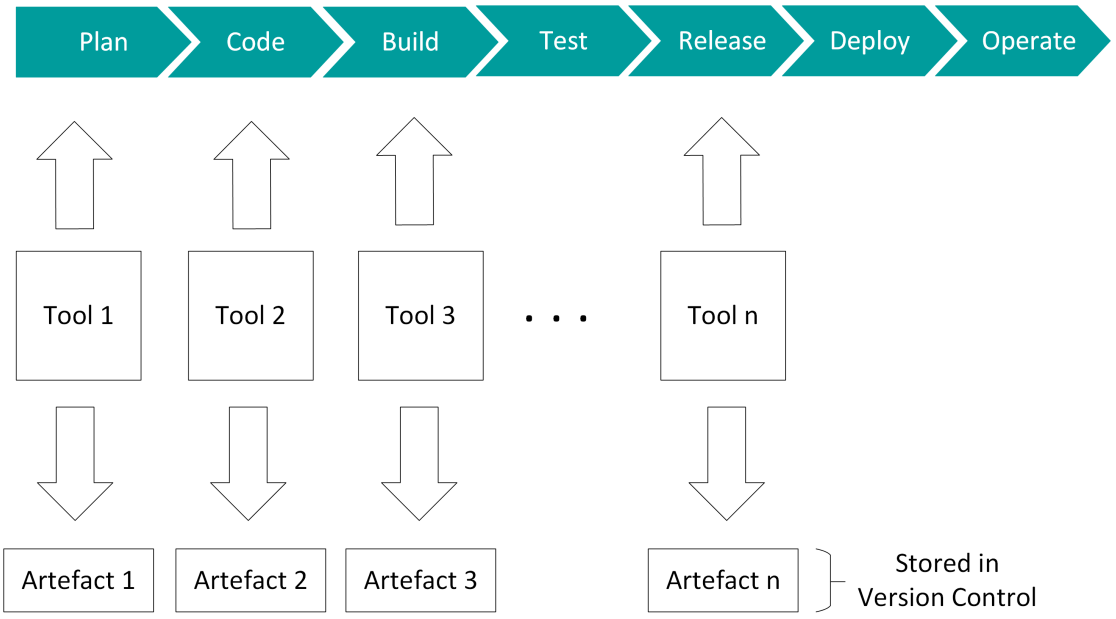


FIGURE 3.3: Stored artefacts by tools for compliance proof

Figure 3.3 illustrates the idea that the tools that aid performing the 4-1 activities across the pipeline store the generated results of the activities (artefacts 1 to n) in a version control system. Since they act as a quality gate, it makes the artefacts dependent, i.e., there's the need to have all the artefacts stored until *artefact n* in order to generate *artefact n*. Thus, guaranteeing that all the artefacts

needed to achieve 4-1 proof of compliance are stored by the end of the product life cycle.

To this extent, the intention is to leverage on the recommendations given in the previous sections by specifying a DevOps pipeline compliant to the IEC 62443-4-1 standard. Thus, making the life cycle secure from the beginning, support transparency and traceability, as well as tackling the trade-off between security and the competition with the speed of digital business by leveraging automation as much as possible.

Chapter 4

Proposed methodology

This chapter is divided into two sections. The first section contains a description of the proposed methodology to enable the development of the proposed concept in the previous chapter. The second section details all the steps taken towards the application of the methodology described in the first section.

4.1 Description of the methodology

In this section, it is detailed all the necessary steps towards the development of the concept proposed in the previous chapter. This section is divided into three subsections, which are ordered as the following:

1. Define and create templates for automation tool support;
2. Create standard specification templates;
3. Observe and analyse the templates;

4.1.1 Define and create templates for automation tool support

Table 4.1 illustrates the defined structure of the templates for automation tool support for each of the 4-1 requirements. The line of the table with the label **ID** corresponds to the identification code of the task, event or gate according to the given 4-1 process models, which identifies its number, as well as the corresponding standard practice. The line **Input** contains the identification number of the necessary file or other activity that triggers that specific task, event or gate. The third line contains its description and will be named as **Task**, **Event** or **Gate** accordingly. **Automation** contains the automation level, which is further detailed in the Section 4.1.1.1. Finally, the **Tool** line contains the identification of the tool that aids with the execution of the task, event or gate according to the criteria detailed in the Section 4.1.1.2.

ID	Identification code of the event or task
Input	File or event input
Description	Description of the Task, Event or Gate
Output	Resulted output
Automation	Automation Level
Tool	Tool identification

TABLE 4.1: Template for automation tool support

4.1.1.1 Classification criteria definitions for automation

In order to identify tool capabilities for all the tasks, events and gates identified in the provided process models, five different possible automation levels were defined. The classification criteria proposed for automation level are the following:

- **Human Task** - For tasks, events and gates that have to be manually performed by a security expert and/or practitioner;
- **Partial** - For tasks, events and gates that can be partially automated but have to be manually completed;

- **Transparency** - For tasks, events and gates to be performed by a human, but based on a tool visualisation of a problem;
- **Complete** - For tasks, events and gates that can be fully automated and performed by a tool;
- **Tool possible** - For straightforward tasks, events and gates for which a tool could be build, but there is currently no tool available.

4.1.1.2 Tool selection criteria

The tool selection is directly dependent to the selected **Automation Level** and has to follow specific criteria. The criteria is defined as described bellow, without following a specific order:

- If a task, event or gate has its **Automation Level** defined as **Human Task** or **Tool Possible** it will be assigned as **"N/A"** as it is not applicable for the specific case;
- It is required to have at least one tool identified for the remaining **Automation Level** values, **Partial**, **Transparency** and **Complete**;
- Give priority to open source over proprietary tools;
- Given that this specification is intended to be used in a pipeline, give priority to tools that easily integrate to **Continuous Integration** tools (e.g., *Jenkins*, *Gitlab CI*).

4.1.2 Create standard specification templates

This phase consists on creating a specification of the entire 4-1 standard in the form of an Excel file that defines the 4-1 practices as a number of templates detailing the automation capability of each 4-1 requirement, as defined in the previous section. The flow of information between the requirements is represented as arrows based

on the process models described in the previous chapter. In order to represent this idea, Figure 4.1 corresponds to an excerpt of the idea. In this specific case it is relative to the **Secure Implementation Review** requirement from the **Secure Implementation** practice.

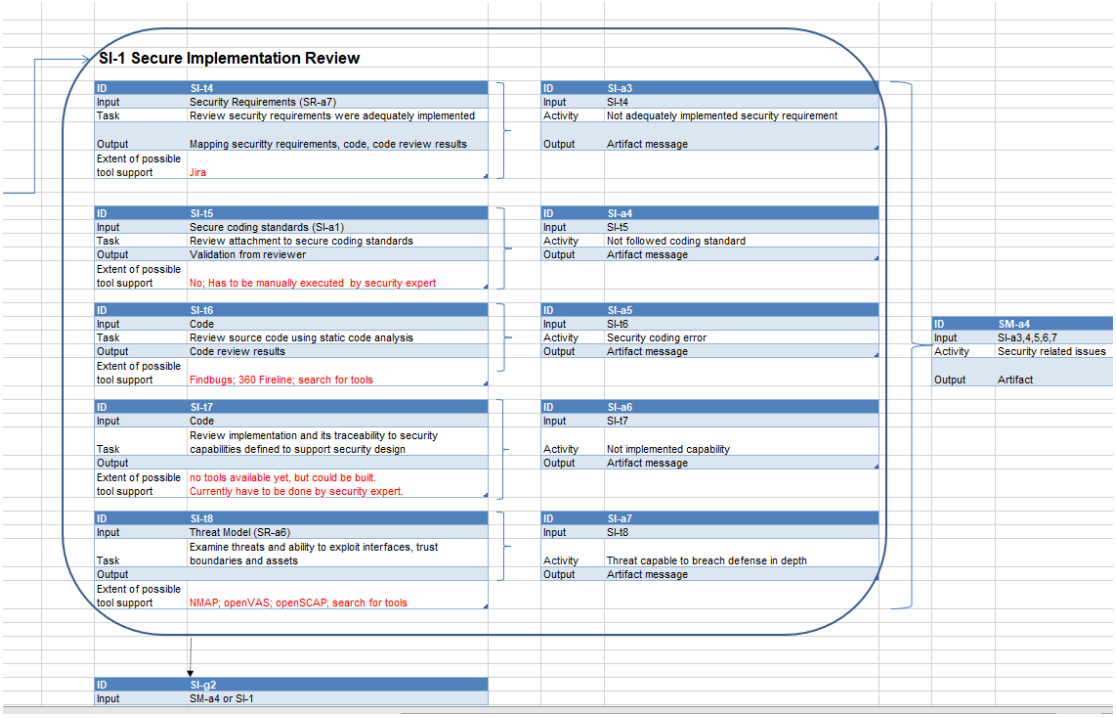


FIGURE 4.1: 4-1 Standard Excel Specification Excerpt

4.1.3 Observe and analyse the templates

Once all the previous steps are executed, this phase consists on observing and analysing the obtained results from the creation of all the automation tool support specifications. For this, it was decided to create a summary of all the automation tool support tables created. Table 4.2 shows the defined template for this summary, which follows a similar structure as the defined templates in the Section 4.1.1.

Practice	Id	Type	Automation	Tool	Creates Artefact
P_X	Id code	Task	Automation Level	Tool name	Yes
P_X	Id code	Event	Automation Level	Tool name	No
P_Y	Id code	Gate	Automation Level	Tool name	No

TABLE 4.2: Table template for summary of the specification

As opposed to the previously defined template, the description is omitted, however, Table 4.2 additionally mentions the **Practice**, the **Type**, which can take the values **Task**, **Event** or **Gate**, and indicates if there is an **Artefact** being created out of that specific task, event or gate.

This table serves the purpose of making it easier to gather information regarding the possibility of automation across all the 4-1 standard and filter according to different levels of criteria. Furthermore, it will allow to answer the research question **RQ1-a** (How much tool support can we provide for 4-1 compliance?).

4.2 Application of the methodology

This section details all the steps towards the application of the defined methodology in the previous section. It is divided in four subsections. Section 4.2.1 explains the process of creating the tables for automation tool support. Followed by the creation of standard specification Excel templates in Section 4.2.2. The challenges of applying the methodology are mentioned in section 4.2.3. Finally, Section 4.2.4 shows an observation and an analysis of the obtained results.

4.2.1 Creation of tables for automation tool support

4.2.1.1 Process model analysis

The first step towards the creation of the tables for automation tool support is the analysis of the provided 4-1 process models. After an initial analysis, it was identified the amount of tasks, events and gates present in the process models, as shown in Table 4.3.

The practice **Security Management** is different from the other practices. Not only this practice occurs throughout the whole product's life cycle, there are artefacts that are created from activities that are different from the previously shown (Tasks, Events and Gates). The majority of these activities and artefacts

Practice	Tasks	Events	Gates	Total
Practice 1	9	21	5	35
Practice 2	12	0	1	13
Practice 3	22	1	4	27
Practice 4	8	1	2	11
Practice 5	16	0	2	18
Practice 6	21	8	5	34
Practice 7	9	2	1	12
Practice 8	10	0	0	10
Total	107	33	20	160

TABLE 4.3: Number of Tasks, Events and Gates identified in the process models

may change over time and are mainly related to organisational documentation, such as "*Organisational roles and personnel responsible*" and "*Personal security qualifications*". These activities and artefacts are highlighted in orange in Figure 4.2, which shows an excerpt of the **security management** practice. For this reason, these activities **are not included** in the scope of this project and **are excluded** from the automation tool support tables.

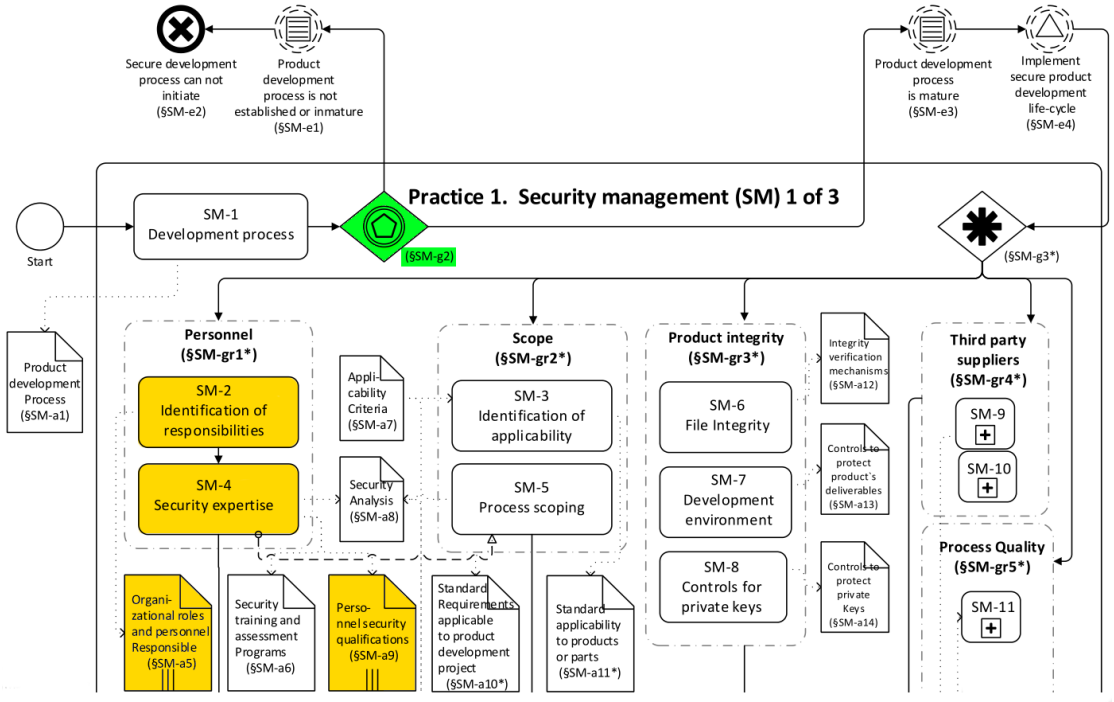


FIGURE 4.2: IEC 62443-4-1 Process models excerpt - Security Management practice

It was also found that some of the gates mainly have the purpose of triggering other activities or making a decision and contain no description. For this reason, a

small textual description was added to the automation tool support templates of the gates to make it more clear. An example of this is gate *SM-g2* in **Security Management** practice, in which there is a need to make a decision whether the product development process is mature or not, as highlighted in green in Figure 4.2. Another example is the gate *SD-g3* in **Secure Design** practice, which serves the purpose of triggering a set of tasks regarding the requirements of **secure design principles** for an interface, as shown in Figure 4.3.

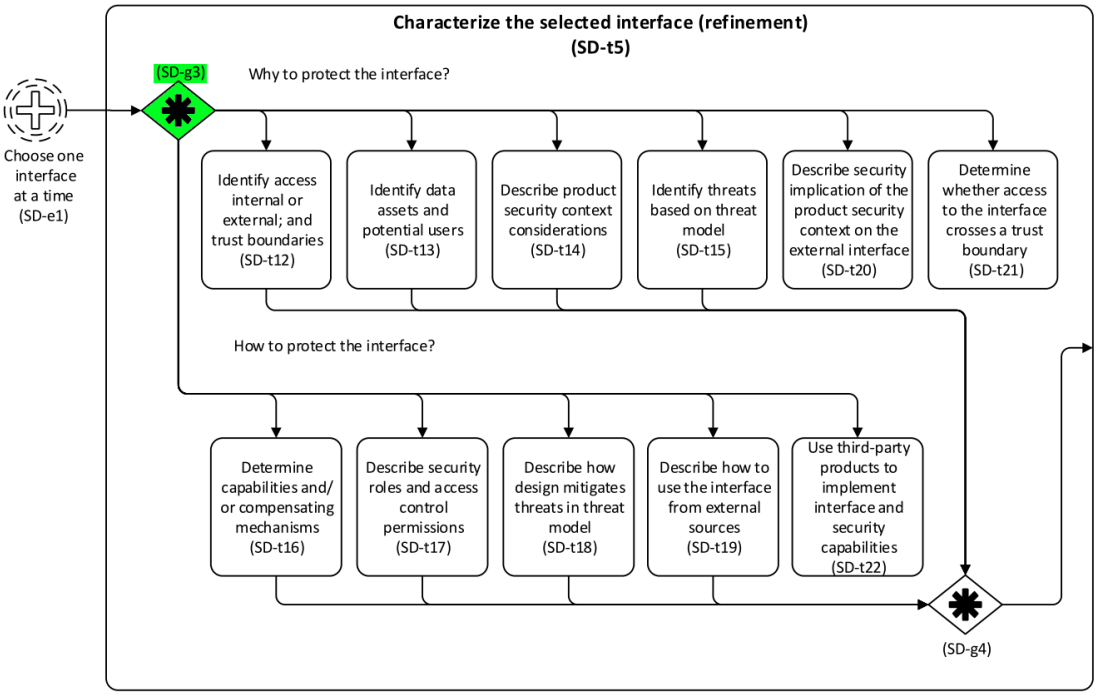


FIGURE 4.3: IEC 62443-4-1 Process models excerpt - *SD-g3* gate is highlighted in green

4.2.1.2 Automation level and tool selection

Subsequently, the creation of tables for automation tool support is processed for each of the identified tasks, events and gates from the Table 4.3. For each of them it was analysed the scope of the practice and requirement it belonged to, the flow of events that surrounded them, as well as the necessary artefacts that serve as input or output.

It was noted that the automation level and the selected tool are dependent variables in the tables, since the classification of the automated level depends on

the capabilities of the identified tool (i.e., a task may seem to be fully automated, however the identified tool can only partially automate it and has to be manually completed).

In order to fulfil the **tool** and **automation** level fields at the table it was followed the following approach:

1. Comprehensively understand the necessary action to fulfil each of the activities, **in particular tasks**, since most of the identified gates and events serve the purpose of triggering other tasks;
2. Look for reliable sources such as (Hsu, 2018; SANS, 2018), as well as systematic reviews on tools for continuous integration, delivery and deployment, such as (Shahin, Babar, & Zhu, 2017) in order to find possible tools to fulfil a certain activity;
3. Investigate the selected tool and check if it fits the activity and select the adequate level of automation;
4. Discuss with company supervisor and security practitioner/expert in order to validate accuracy and consistency of the selection, in particular automation level selection.

Table 4.4 is an excerpt of the automation tool support, which is a result of the above sequence of actions. It corresponds to a task that belongs to the requirement **secure implementation review** from the practice 4, **Secure Implementation**. The assigned tool for this task was *Jira*, which is a tool that is used to aid with planning, tracking and managing agile software development projects. The outputs from this task can also forward an artefact message in case a security requirement is not adequately implemented, which will result in triggering an event to alarm that security-related issues were detected. This will be further detailed in the following section.

ID	SI-t4
Input	Security Requirements (SR-a7)
Task	Review security requirements were adequately implemented
Output	Mapping security requirements, code, code review results
Automation	Partial
Tool	Jira

TABLE 4.4: Automation tool support excerpt

In order to get a better understanding of the **transparency** automation level, the following tables, Table 4.5 and Table 4.6 correspond to excerpts of tasks evaluated with this automation criteria.

As illustrated in Table 4.5, *SR-t12* corresponds to a task in which a security practitioner or expert has to accept security requirements submitted by other practitioners into a repository. The input received in this task is a gate in which a requirement is checked for its clarity and validity. This task was classified as **transparency** since the tool *Jira* helps the author of the decision, with the visualisation of the requirements.

ID	SR-t12
Input	SR-g1
Task	Approve security requirements
Output	Practice 3 - SD
Automation	Transparency
Tool	Jira

TABLE 4.5: Transparency tool support excerpt - SR-t12

Task *DM-t3*, represented in Table 4.6, is also classified as **transparency**, as *Jira* helps the practitioner with the visualisation and tracking of the issues. The input received in this task corresponds to a notification of the given issues. The resulting output is the placement of the issue tracked into a repository and the trigger of an event that starts a set of tasks to investigate the issue.

In order to get an overview of the **tool possible** automation level, the following tables, Table 4.7 and Table 4.8 correspond to excerpts of tasks evaluated with this automation criteria.

ID	DM-t3
Input	DM-t2
Task	Track security-related issues
Output	DM-a2; DM-e1
Automation	Transparency
Tool	Jira

TABLE 4.6: Transparency tool support excerpt - DM-t3

As illustrated in Table 4.7, *SR-t8* corresponds to a task in which the security requirements have to be identified based on the security level-capability (SL-C). SL-C is used to determine the overall security level of a component or system if adequate information is provided (for example, if a component contains a capability security level of 3, it means that the component or system includes the capability to achieve security functions required for capability security level 3 according to the IEC 62443-4-2 standard (for components) (Blumano, 2018). The task receives as an input, the required SL-C of the component from the previous task *SR-t7*, the threat model and the IEC 62443-4-2 standard (4-2). The 4-2 standard provides detailed requirements for technical control system components according to the selected level of SL-C associated with the seven foundational requirements, which are identification and authentication control, use control, system integrity, data confidentiality, restricted data flow, timely response to events and resource availability (IEC, 2019).

Since the requirements of the components are detailed in the 4-2 standard for the seven foundational requirements according to the desired SL-C level, different components can have the same or similar requirements. For this reason, this task was assigned with **tool possible** in the automation criteria.

Table 4.8 corresponds to a task in which a set of appropriate practices must be selected for a product, artefact *SD-a2*, based on a given list of industry accepted secure design best practices , artefact *SD-a1*. Since a list of practices is provided for this task, it was classified as tool possible due to the likelihood of similar products receiving the same set of practices as an output of this task.

ID	SR-t8
Input	SR-t7; SR-a3 (IEC 62443-4-2 Standard); SR-a6 (Threat model)
Task	Identify security requirements based on capability security level
Output	
Automation	Tool Possible
Tool	N/A

TABLE 4.7: Tool possible excerpt - SR-t8

ID	SD-t1
Input	SD-a1 (Industry accepted secure design best practices)
Task	Determine appropriate secure design best practices for the type of product
Output	SD-a2 (Selected secure design best practices)
Automation	Tool Possible
Tool	N/A

TABLE 4.8: Tool possible excerpt - SD-t1

The IEC 62443-4-1 automation tool support tables set depicts the **complete 4-1 Standard**. For documentation purposes, the tables set is listed in Table 4.9, and the tables for each practice are shown in Appendix B.

Practice	**Table**
Practice 1 - Security Management (SM)	B.1
Practice 2 - Security Requirements (SR)	B.2
Practice 3 - Secure Design (SD)	B.3
Practice 4 - Secure Implementation (SI)	B.4
Practice 5 - Security verification and validation testing (SVV)	B.5
Practice 6 - Management of security related issues (DM)	B.6
Practice 7 - Security Update Management (SUM)	B.7
Practice 8 - Security Guidelines (SG)	B.8

TABLE 4.9: Complete list of automation tool support tables, sorted by practice

4.2.2 Creation of standard specification templates

Once all the tables for automation tool support are completed, it is possible to create the standard specification templates. This phase consists in merging sets of tool templates that belong to the same requirement from each of the 4-1 practices according to the provided process models. This way it is possible to see the flow

of information and the sequence of actions that need to be performed in order to fulfil a given requirement, as well as all the necessary artefacts that serve as input and output.

Figure 4.4 is an excerpt of the standard specification; in this case it corresponds to the **secure implementation review** requirement from **practice 4**, **secure implementation**. In order to fulfil this requirement, there's the need to execute five tasks, which are:

- **SI-t4** - Review security requirements were adequately implemented
- **SI-t5** - Review attachment to secure coding standards
- **SI-t6** - Review source code using static code analysis
- **SI-t7** - Review implementation and its traceability to security capabilities defined to support security design
- **SI-t8** - Examine threats and ability to exploit interfaces, trust boundaries and assets.

As mentioned in the description of *SI-t4* in the previous Section, each of these tasks shall forward an artefact message in case a security related issue is found during the reviewing process. It will then create the artefact labelled as *SM-a4*, security related issues. Once all the tasks from the requirement were executed, the gate labelled as *SI-g2*, will check if the artefact *SM-a4* was created to verify if any security related issue was found. In case there aren't any issues found, it will proceed to the **practice 5**, **security verification and validation testing**. If there was any issue found, it will trigger the event *SM-e6*, security related issues detected, which will then trigger a set of tasks from **practice 6**, **management of security-related issues**, in order to review and fix the issue.

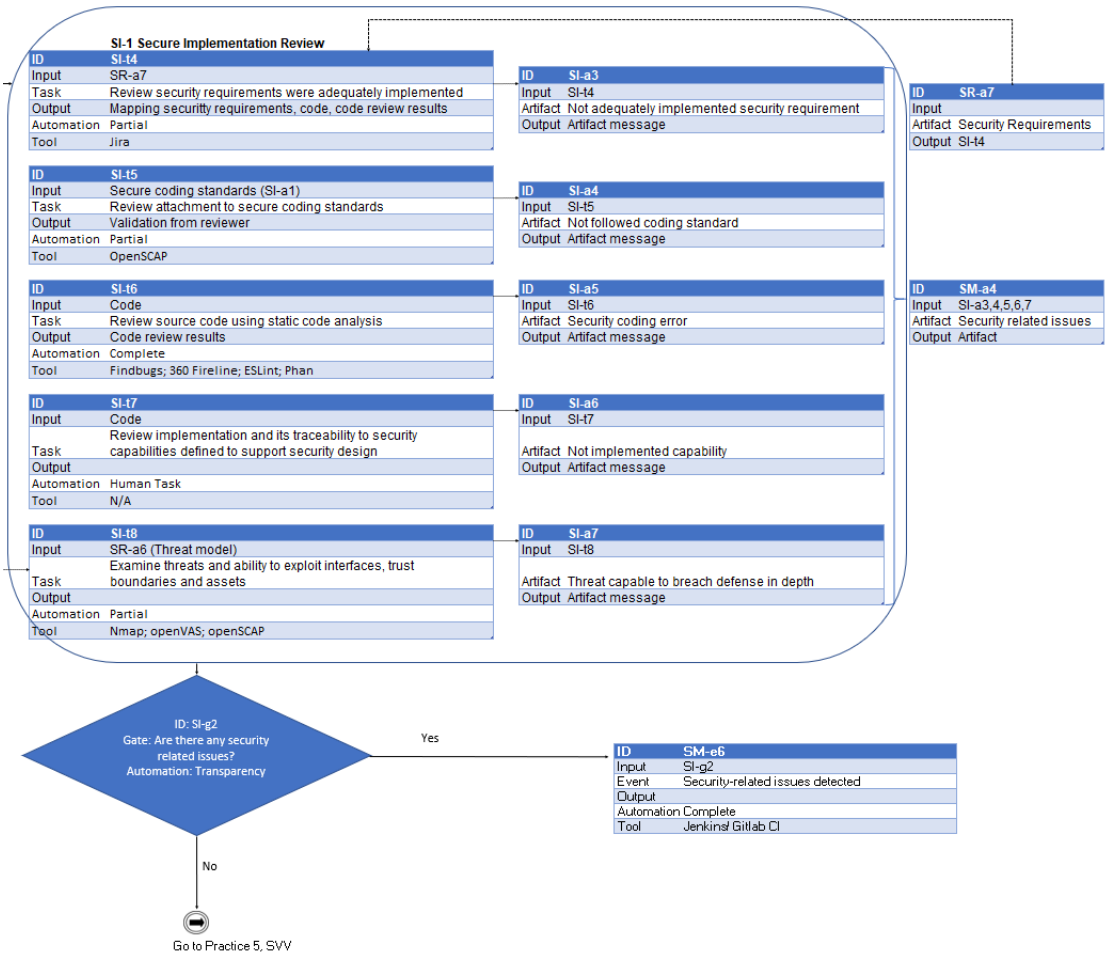


FIGURE 4.4: Standard specification, secure implementation review requirement

4.2.3 Challenges

During the execution of the application of the methodology, several challenges emerged, in particular during the phase of the creation of the automation tool support tables. The challenges are essentially related to the understanding of the given process models and the tool selection to fulfil each of the activities. The emerged challenges are as follow:

- As previously mentioned in the Section 4.2.1.1. some of the activities, **in particular gates**, contained no description, thus its purpose had to be often speculated;

- There was an occasional unawareness of existing tools to fulfil a particular activity, due to the high diversity regarding the fields of interest present in the product development life cycle;
- It was often found solely a tool to fulfil a particular activity;
- There were cases where the description of an activity wasn't clear in the process models, leading to a lot of possible actions or a set of actions to fulfil it, as well as a high variety of tools selected. An example of this is the task *SVV-t4* shown in Table 4.10, which is described as "Conduct functional testing of security requirements". In this particular case, the needed set of actions will highly depend on the product and type of tests needed;
- Once all the tool support templates were completed, a total of **70** different tools were identified. Due to the limitations imposed in the workplace it was not possible to test every single one in order to verify their capabilities of fulfilling the activities and automation level they were assigned to.

ID	SVV-t4
Input	SVV-a7 (Security Requirements)
Task	Conduct functional testing of security requirements
Output	
Automation	Complete
Tool	**Unit tests**: JUnit; Mocha; xUnit; **Smoke tests**: ZAP Baseline scan; nmap **Security Acceptance testing**: BDD-Security; GauntIT; Mittn **Infrastructure Compliance Checks**: inSpec; HubbleStack; **Cloud Compliance**: Cloud Custodian; Compliance Monkey

Table 4.10: Tool support excerpt - *SVV-t4*

4.2.4 Observation and analysis of results

Once the application of the methodology as described in Sections 4.2.1 and 4.2.2 was concluded, it was possible to gather data regarding the possibility of automation of the IEC 62443-4-1 standard.

Figure 4.5 depicts the automation tool support obtained for the IEC 62443-4-1 standard according to the automation criteria defined in the Section 4.1.1.1.

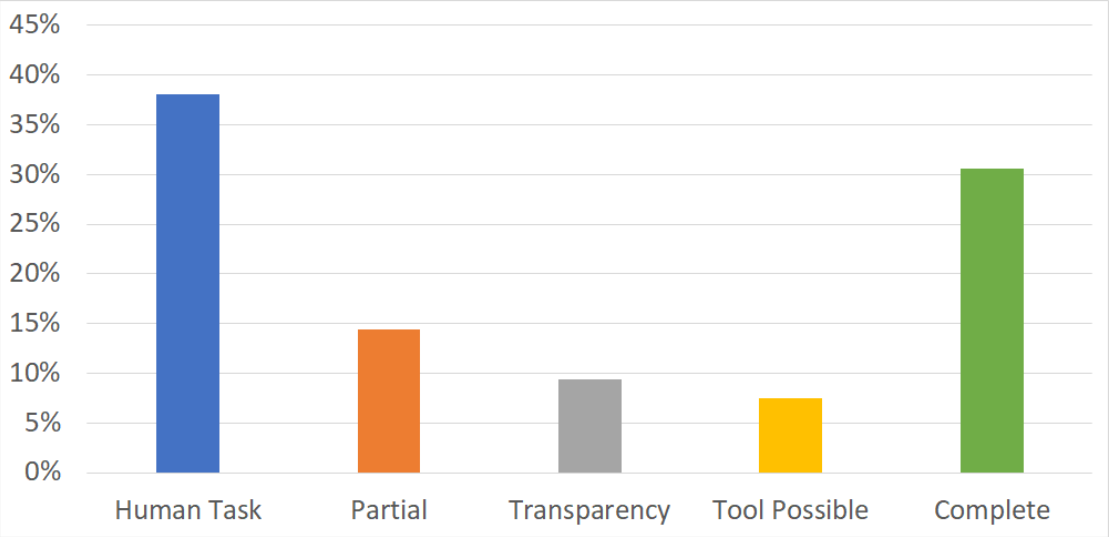


FIGURE 4.5: Obtained IEC 62443-4-1 automation tool support

It is possible to initially observe that **human task** is the classification that is the most predominant, followed by **complete** and **partial**, with **transparency** and **tool possible** being the least identified among the activities analysed in the standard process models.

As previously stated, the 4-1 standard provides descriptions of requirements to enable secure product development life cycle for products in an ICS environment, which need to go through extensive security activities that frequently need to be performed by humans, as well as documented. Hence, **human task** being the most prevalent activity identified in the process models.

Figure 4.6 depicts a more detailed view on the obtained automation tool support for the 4-1 standard, in particular the number of activities identified for each of the automation classification criteria for each of the 4-1 standard practices.

After observing Figure 4.6 it is possible to make the following initial observations:

- Surprisingly, **practice 1**, **security management** contains the highest amount of activities classified as **complete** regarding the automation tool support,

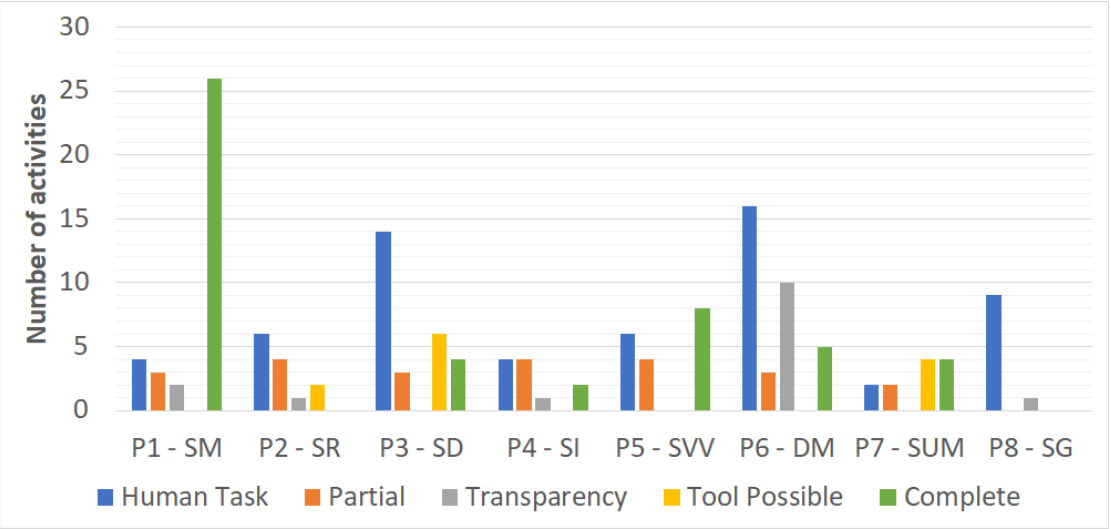


FIGURE 4.6: Obtained IEC 62443-4-1 automation tool support per practice

which initially seemed very unlikely, due to the practice being referred as **security management**. Taking into account the scope of the practice, as well as the analysis of the process models described at the Section 4.2.2.1 from this Chapter, it is possible to observe that this happens due to the high amount of events and gates that are present in the process models that often serve the purpose of triggering other activities from all of the practices during the product development life cycle;

- **Practice 5**, **security verification and validation testing**, contains a balanced amount of activities classified as **human task**, **partial** and **complete**. This happens due to the variety of activities present in the practice, which include the execution of several tests with the aid of tools, as well as the analysis and documentation of the generated results that often require human effort;

- It was expected for **practice 6**, **management of security-related issues**, to contain the majority of the activities classified as **transparency** due to its purpose of tracking and handling security related issues, which often require decision making based on the visualisation of a problem;

- **Practice 8**, **security guidelines**, has the main purpose of enforcing the creation of documentation for the end user of the product, hence contains no automation and has mostly **human tasks**.

Chapter 5

Evaluation

This chapter details the steps taken towards the evaluation of the present work. The possible automation of the 4-1 standard and its integration into a DevOps pipeline are evaluated through a qualitative study. The evaluation was carried out through the aid of **interviews that involved 7 expert practitioners** from the fields of security compliance and agile software engineering. This Chapter is divided in three sections. Section 5.1 starts by explaining the evaluation goals. The design of the evaluation is followed in Section 5.2, detailing the interviewee selection, the structure and flow of the interviews. Section 5.3 analyses the data collected from the interviews to evaluate the proposed solution for the possibility of building a 4-1 standard compliant DevOps pipeline. The possible threats to the validity of our study are described in Section 5.4. Finally, Section 5.5 presents an overview of the evaluation of the proposed solution.

5.1 Objective

The objective of this phase is to answer the research questions mentioned in the first chapter:

- Determine the **precision** of the defined automation criteria and the **feasibility** of the suggestion for automation of the 4-1 standard in the pipeline.

This will conclude **RQ1** on the suitability of our approach to build a 4-1 compliant DevOps pipeline.

5.2 Design

5.2.1 Overview

(Hevner, March, Park, & Ram, 2004) suggests the use of a descriptive evaluation method to build a convincing argument for an artefact's utility, as well as the study of an artefact in a restricted environment, for qualities, such as usability, accuracy and fit within the organisation. It is also mentioned by (Hevner, March, Park, Ram, 2004) that the use of descriptive methods of evaluation shall be used exclusively for innovative artefacts for which other evaluation methods are not appropriate. This is the case of the artefact we propose, since there is no other with similar characteristics to compare with. Furthermore, it is meant to explore the practitioners' opinion on the suitability and feasibility of our suggestion, making interviews a suitable technique for the evaluation (Shull, Singer, & Sjøberg, 2007).

A total of **7 expert practitioners**, that regularly take part in software development projects that involve security compliance, participated in the interviews. The evaluation goes in accordance with techniques mentioned in current security research and comparable studies, (Ben Othmane, Jaatun, & Weippl, 2017), that have a restricted environment, contain sampling sizes of 5 to 11 practitioners, in which interviews are used as an evaluation method that take half an hour to an hour.

5.2.2 Interviewees

The interview participants are experienced professionals, with different levels of knowledge and expertise in the field of study of the present work. In order to document this, the participants were asked about their awareness in the 4-1 standard,

DevOps, DevSecOps and security tools. The participants were able to select their level of expertise in the topics according to the classification levels of **Beginner, Medium, Advanced** and **Expert**, as depicted in the introductory questions of the questionnaire (see Appendix C). For anonymity reasons, the participants' names and roles in the company were omitted.

Table 5.1 show the interviewees' selected levels of expertise at the beginning of the interview, according to the aforementioned classification levels. It is possible to observe that the participants have different levels of knowledge and expertise in the topics and that none of the participants has classified himself as an **expert** in any of the given topics.

-	Beginner	Medium	Advanced	Expert
4-1 Standard	2	3	2	0
DevOps	2	2	3	0
DevSecOps	2	2	3	0
Security tools	1	3	3	0

TABLE 5.1: Interviewees' selected level of knowledge and expertise in the different topics

5.2.3 Interview

As a preparation for the interviews, it was selected one task for each of the defined automation classification criteria defined in Section 4.1.1.1 from the 4-1 standard **practice 2, security requirements** and **practice 5, security verification and validation testing**. Subsequently, these sets of process models were printed, as well as the corresponding standard specification templates. In addition, it was printed the high level specification of the 4-1 standard DevOps compliant pipeline as represented in Figure A.1 (see Appendix A). The selected tasks used for the evaluation phase correspond to the following:

- **SR-t7** - Document the required capability security level (SL-C) of the product;
- **SR-t8** - Identify security requirements based on the capability security level;

- **SR-t12** - Aprove security requirements;
- **SVV-t4** - Conduct functional testing of security requirements;
- **SVV-t13** - Test dynamic runtime resource management.

Interviews involved one participant and two interviewers, the master thesis author and the company's supervisor. A questionnaire (see Appendix C) was filled[1] as the interviews were taking place. The interviews had the following sequence:

1. **Introduction** - The participants were informed about the goal and the flow of the interview. Furthermore, the participants were asked to classify their level of knowledge and expertise in the different fields of study, according to the **introductory questions** depicted in the questionnaire.
2. **Part 1: 4-1 Process models and high level specification** - First, the participants were introduced to the 4-1 standard and its process models developed by (Moyon et al., 2018), with the aid of the printed versions of the practice 2 and 5 of the 4-1 process models. Subsequently, they were introduced to the proposed solution with the aid of a printed version of the high level specification presented in Figure A.1 (see Appendix A). Later, the participants answered the questions in the part 1 of the questionnaire.
3. **Part 2: Automation tool support** - Participants were introduced to the standard specification templates that included the aforementioned tasks. The participants were given a chance to interact with the specification templates and analyse the aforementioned tasks. Afterwards, they proceeded to answer the questions in order to evaluate the selected automation level and tools to aid with each of the tasks, as depicted in the 2nd part of the questionnaire.
4. **Part 3: Overview and feedback** - In order to get an overview of the obtained results, the participants were introduced to the graphics illustrated

[1]Due to company's confidentiality policies, the answered questionnaires were excluded from this report

in figures 4.5 and 4.6 from Section 4.2.4 of the previous Chapter. Later, the participants answered the final questions from the 3rd part of the questionnaire.

The interviews took an average of one hour, with Figure 5.1 illustrating the flow and the duration of the different parts of the interview.

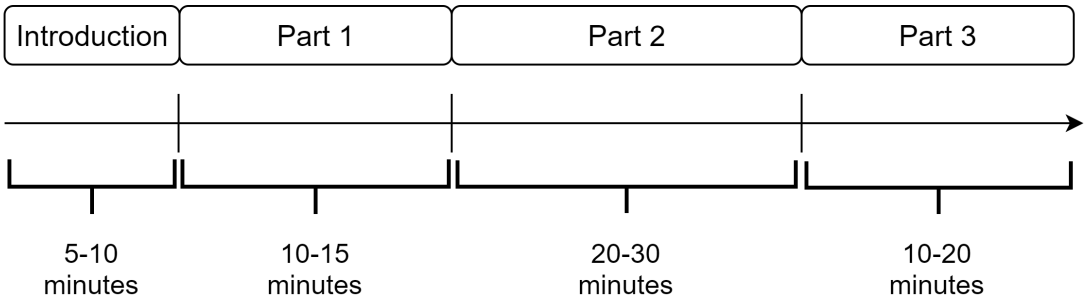


FIGURE 5.1: Flow of the conducted interviews

5.3 Analysis of the evaluation results

This section puts together the results obtained from the interviews described in the previous section. The analysis of the results are divided in three different subsections, corresponding to the three different parts of the questionnaire (see Appendix C).

5.3.1 Classification criteria

The first part of the questionnaire involved two questions regarding the defined classification criteria for automation in Section 4.1.1.1. Initially the participants were asked about the preciseness of the classification criteria, later they were asked if the criteria would be helpful for building a security compliant pipeline.

- **Preciseness of the classification** - The majority of the participants agreed that the defined classification criteria for automation was precise, however,

some of the participants had initial doubts about the meaning of "transparency" and "partial" definitions. It was said by Interviewee 4 that "Partial is not precise as it is hard to measure tool work and human effort in terms of percentage". Interviewee 6 argued about the meaning of "complete automation", stating that there's always the need to visualise the results. Only interviewee 5 disagreed with the classification criteria, mentioning that "partial" and "transparency" classification criteria correspond to the same classification. Interviewee 5 also said: "You will always have visualisation and it is not relevant for tool automation".

- **Applicability for pipelines** - Three of the participants agreed that the criteria could be helpful for building a pipeline. Interviewee 3 said "Yes, because you could see where you can use a tool for automation". It was mentioned by Interviewee 5 that it would depend on the kind of pipeline and its work environment. However, the other participants highlighted that the criteria would be better for evaluating a pipeline. Interviewee 7 said "Could be helpful to evaluate a pipeline, not to build it". A similar answer was given by interviewee 2, stating that it would allow to visualise how much is possible to automate.

5.3.2 Automation and tool selection

The second part of the questionnaire contained seven questions for the participants to evaluate the selected tool and automation level for each of the tasks mentioned at the section 5.2.3. The goal of this part of the interview is to gather enough information to understand if our suggestion for automation of the 4-1 standard activities is feasible or not. Table 5.2 contains the automation classification of the evaluated tasks in this part of the interview.

- **Correctness** of the automation selection and **feasibility** of the suggestion for 4-1 automation - Overall, the participants agreed on the chosen automation criteria defined for the selected 4-1 tasks. However, the participants

Task	Automation classification
SR-t7	Human Task
SR-t8	Tool Possible
SR-t12	Transparency
SVV-t4	Complete
SVV-t13	Partial

TABLE 5.2: Automation classification of the evaluated tasks

discussed about the selection for the task defined with the level of "complete automation".

- ***SVV-t4*** - This task made the participants argue once again about the meaning of "complete automation", as the task would require some human effort in configuration in order to be automated. It was said by interviewee 8 " Yes, if defining the test cases are not in the task.". As previously mentioned in Section 4.2.3, the description of this task in particular isn't clear enough in order to have a well defined action to accomplish it. Interviewee 4 mentioned "I'm not sure if it is complete, it will depend on the product and feature". Interviewee 5 disagreed on the selection, mentioning that it "should be partial depending on the components and type of test";
- ***SVV-t13*** and ***SR-t12*** - These two tasks didn't leave much room for discussion. Every participant agreed on the selection for the task *SVV-t12*, apart from Interviewee 5 that had previously argued on the different between "partial" and "transparency" classifications, and it was acknowledged the use of the tool *Jira* to accomplish the task. As for the task *SVV-t13*, the participants agreed on the automation level selection, even though most of them confessed not to be aware of the selected or existent tools to perform the task;
- ***SR-t7*** - Only 2 of the participants disagreed with the automation selection in this task. It was said by Interviewee 5 that "I think there's a Microsoft tool that exists in which you put your product and it returns the SL-C.". However, three of the participants that agreed stated that it might be tool possible. Interviewee 6 agreed and proceeded to argue

that maybe it could be tool supported with the help of machine learning. It was agreed and then argued by interviewee 7 that "maybe this task could be transparency".

- ***SR-t8*** - All but one participant agreed with the selected level in this task. The participant mentioned that "from the point of view, the requirements are listed and filtered in an Excel sheet", meaning that the task is supported by a tool. Some of the participants that agreed mentioned that the task could reach the automation levels of partial or transparency. As Interviewee 4 mentioned, "I agree with the selection, but this would never be complete".

- As the participants have different levels of knowledge and expertise in the different fields of study of the present work, they have different awareness of the selected tools to fulfil the 4-1 standard activities. This was true, in particular, for the tasks ***SVV-t4*** **and** ***SVV-t13***, as all the participants were aware of the capabilities of the tool *Jira* and its capabilities of fulfilling the task ***SR-t12***.

  - One of the questions from the interview was to ask for other tool recommendations that may fulfil the evaluated 4-1 standard tasks. Some of the participants with higher knowledge in security tools, in particular interviewees 5, 6 and 7, recommended tools to fulfil the tasks ***SVV-t4*** and ***SVV-t13***. The capabilities of the recommended tools to aid the fulfilment of the task in question was discussed between the participant and the interviewers. This led to the addition of some tools to the automation tool templates.

5.3.3 Overall feedback

In the final part of the interview, the participants were given an overview of the obtained results and answered three questions. These questions' goals were to

verify if the results have met their expectations and to receive an overall feedback about our proposed solution.

- The participants mentioned that they expected slightly more automation and less human tasks, as they observed Figure 4.5 from the previous chapter. Interviewee 5 said "It looks ok, I just expected less human tasks and more activities marked as complete". It is also important to note that the interviewees have different knowledge on the standard, as it was said by Interviewee 6 "I expected a little bit more *Tool Possible*, however I need to know the standard better".

- Despite the participants' expectation on higher automation, some of the participants mentioned that the results looked more realistic once introduced to Figure 4.6, with a more detailed distribution of activities and their assigned automation level throughout the standard. However, all of the participants were surprised with the fact that **practice 1**, **security management** contained the most activities labelled as "complete" automation, as previously mentioned in section 4.2.4 of the previous chapter. Interviewee 2 even mentioned "In general it meets the expectation, however the level of automation in practice 1 is surprising". Interviewee 3 has also said that "It is surprising that practice 1 has so much automation. The rest seems to be more realistic".

- The participants didn't comment on each of the 4-1 standard practices. Nevertheless, three participants said that the results obtained for practice 6, ***management of security-related issues*** were the expected ones, with Interviewee 2 saying "Transparency in practice 6 is an obvious match".

The last question of the interview led to more open discussions in which the participants referred benefits and drawbacks of our solution, as well as suggestions and other possible use cases for our approach.

Three of the participants (Interviewee 3, 5 and 6) have suggested benefits regarding the tools. They have pointed out benefits for practitioners regarding the

tool suggestion and task automation, as well as other benefits for higher management positions.

- It was mentioned that it is a good source of information to verify tools and its capabilities, as well as verifying which tasks can be automated. It was mentioned by Interviewee 6 that "Ignoring the 4-1 standard, it would be a good source of tools to fulfil the tasks.", even highlighting "Finding the right tool for the right task". Interviewee 3 highlighted that the tooling information would be good to recommend tools to customers;

- The development of databases was suggested by Interviewee 5. The interviewee mentioned multiple uses cases, from tool capabilities to automation classification of tasks, enabling practitioners to discover new tools and to spot replacement tools for the same task. Interviewee 5 also mentioned the classification of the tools in terms of automation level and suitability for people building a pipeline;

- Other benefits were pointed out, with Interviewee 2 saying "Would say it is helpful to motive other business units by showing the results". Interviewee 7 also mentioned that it is good to make an analysis of what is available. Additionally, Interviewee 4 mentioned that it is good to get a better understanding of the challenges for development and issues when it comes to implementing a standard.

- A few drawbacks of our approach were pointed out during this part of the interview. Interviewee 2 mentioned that technical practitioners might argue about the meaning of automation. It was mentioned by interviewee 4 that the statistics shown are more useful for higher management and that the tool support overview is more granular for the people who use it for daily work. Interviewee 5 pointed out that the work is focused on a specific model of the standard, and that it may influence the obtained results.

5.4 Threats to validity

This section reveals possible threats to the validity of our study.

Representativeness: In order to avoid bias, the participants' selection and interview structure was reviewed by the company supervisor. Additionally, the participants' background and suitability to evaluate our suggestion was confirmed at the beginning of the interviews. The sampling size of 5 to 11 practitioners goes in accordance with current security research and comparable studies that have a restricted environment (Ben Othmane et al., 2017).

Experimenter bias: A particular case for interviews is that the presence of the researcher may affect the behaviour of the interviewees, and therefore, may affect the validity of the data provided by subjects (Shull et al., 2007). To avoid this, the participants were initially informed about the nature of the research, as well as how data was collected. Additionally, the interviewers (the master thesis' author and company supervisor) remained in silence as the participants answered the questions in order to maintain neutrality.

Confirmation bias: The master thesis' author may have a tendency to make false interpretations to confirm hypotheses and overlook information that argues against it. In order to avoid this, the project was reviewed by an academic supervisor and a company supervisor.

5.5 Overview

The present work has shown evidence on the possibility of providing tool support for the IEC 62443-4-1 standard and the specification of a DevOps pipeline compliant to the 4-1 standard. Different aspects of our proposed solution were evaluated in the interviews, in order to investigate the adequacy of the proposed solution. First, the majority of the participants found the classification criteria to be precise. Then, the majority of the participants agreed with the classification of the

selected 4-1 activities with the defined automation classification criteria. However, it was not possible to obtain a definitive answer to **RQ1**, based on the adequacy feedback received in the interviews, as the answer of the interviewees were divided between the suitability of our proposed solution to build or to evaluate pipelines.

Chapter 6

Conclusion and Future Work

6.1 Conclusion

The present work has proposed a concept for a structured and systematic integration of security activities into a DevOps pipeline. The basis for this concept is the integration of the current IEC 62443-4-1 (4-1) standard, that describes secure product development in industrial control systems, and its integration into a Continuous Integration/Continuous Delivery pipeline specification. Thus, pursuing the capability of both secure agile development and security compliant agile software engineering.

The main research domains addressed to achieve the proposed concept are DevOps and pipeline specifications, DevSecOps and security compliant product development. A review on the research domains allowed an understanding of the current challenges and recommendations for secure agile software engineering, as described in the chapter 2. The present work intended to leverage on these recommendations and a key contribution of the 4-1 standard, by (Moyon et al., 2018), that described the standard in the form of BPMNs.

Once the research domains were reviewed, templates for automation tool support were defined, with the goal of supporting the process models created by (Moyon et al., 2018) and discussing which 4-1 activities can be automated with

the help of tools and which cannot. Additionally, it was defined an automation classification criteria in order to identify tool capabilities for all the activities present in the 4-1 standard.

After the analysis and classification of all the 4-1 activities according to the defined automation classification criteria in the section 4.1.1.1, it was possible to answer the following question:

RQ1-a - How much tool support can we provide for 4-1 compliance?

The classification resulted in a total of **54,4%** of the 4-1 activities being identified as tool supported. This percentage accounts for 30,6% of the activities being classified as **complete**, 14,4% classified as **partial**, and 9,4% classified as **transparency**. The remaining 45,6% of the 4-1 activities were classified as not tool supported, with a total of 38,1% of the activities being classified as **human task**, and a total of 7,5% as **tool possible**.

Our proposed solution was evaluated through a qualitative study, which involved interviews with 7 expert practitioners from the fields of security compliance and agile software engineering. The main goal of the evaluation was to answer the following question:

RQ1 - Is it possible to create a DevOps Pipeline that follows a systematic way to security compliance with IEC 62443-4-1 standard?

The present work shows evidence that it is possible to provide tool support for the IEC 62443-4-1 standard and to specify a DevOps pipeline compliant to the 4-1 standard, since we proposed a concept to develop it. In order to investigate if the proposed DevSecOps pipeline is adequate, different aspects of our proposed solution were evaluated by the interviewees. To begin with, the majority of the participants found the classification criteria to be precise. Later, the majority of the participants agreed with the classification of the selected 4-1 activities with the defined automation classification criteria. However, it was not possible to obtain a final answer, based on the interviews to the adequacy issue, as the answers of

the interviewees were divided between the suitability of our proposed solution to build or to evaluate pipelines.

6.2 Future work

One of the next steps to fully support our hypothesis is to perform more evaluation sessions to further investigate the adequacy of the proposed solution to a DevSecOps pipeline and to perform a demonstration of the implementation of a pipeline aided by our 4-1 tool support templates.

As mentioned in section 4.2.3, it was not possible to test every single tool in order to verify their capabilities of fulfilling the activities and automation level they were assigned to, as it would require longer project execution time. Hence, it would be necessary to perform additional review iterations with expert practitioners on the 4-1 tool support templates to guarantee its preciseness and quality.

In regard to the tasks classified as *tool possible*, it would be important to have further discussions with expert practitioners in order to verify if the tasks assigned with this classification are indeed straightforward for a tool to be built. This would positively contribute for tool support of, both the execution of the task itself, and the 4-1 standard automation.

Additionally, it would be positive to leverage on the suggestions given by expert practitioners during the evaluation of the present work. Particularly, make a mapping of tools by 4-1 standard practice or by tool capabilities, in order to enable practitioners to easily discover new tools or find a tool replacement for a specific task.

Appendices

Appendix A

High-level 4-1 compliant pipeline specification

This part of the appendix presents the initial high-level specification of the 4-1 compliant pipeline specification, as mentioned in Section 3.1.

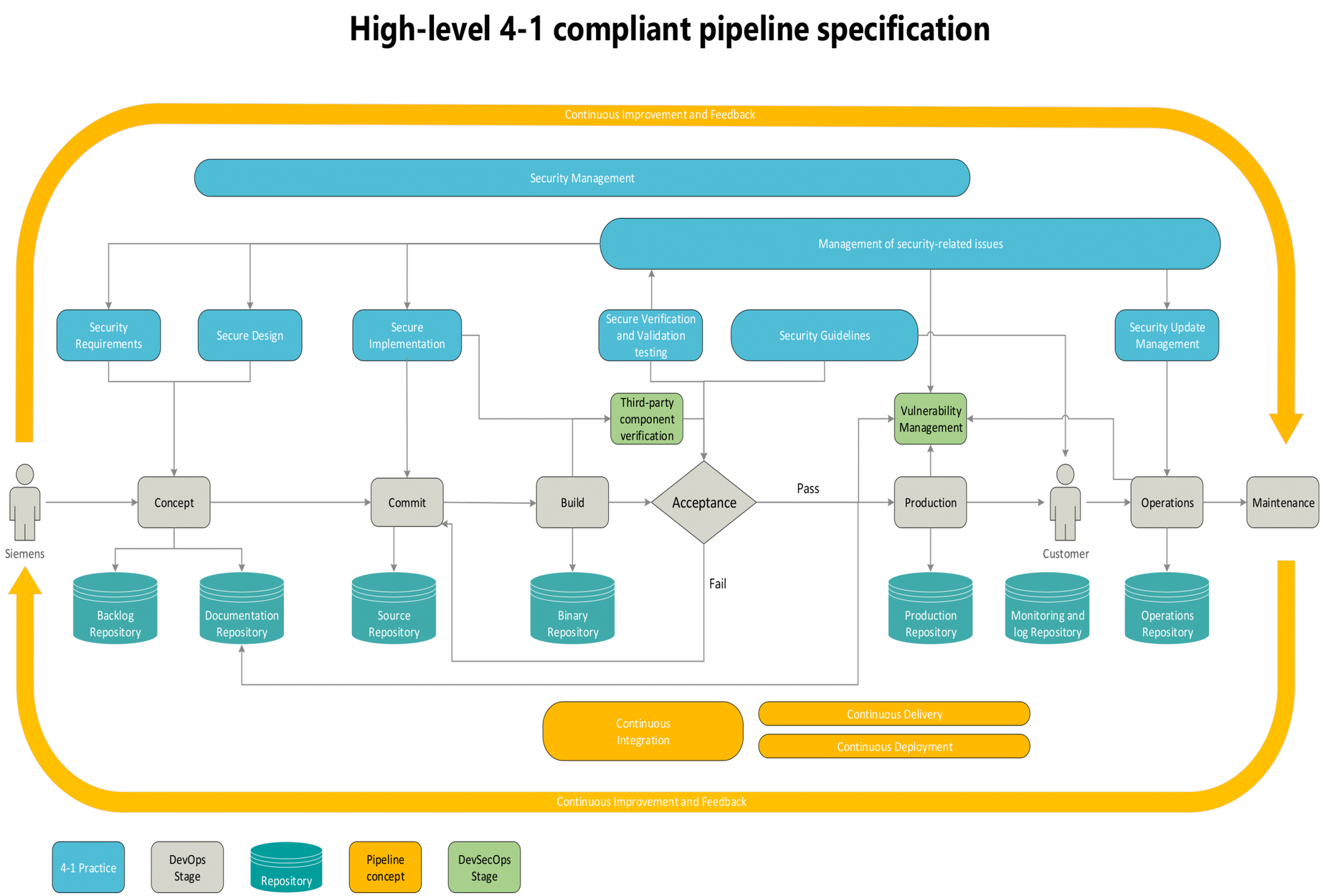


FIGURE A.1: High-level 4-1 compliant pipeline specification

Appendix B

Automation tool support summary tables

This part of the appendix presents the automation tool support summary tables, as mentioned in Table 4.9 from the Section 4.2.1.2.

B.1 Practice 1 - Security Management

Practice	Id	Type	Automation	Tool
P1	SM-e1	Event	Complete	Jenkins; Gitlab CI
P1	SM-e2	Event	Complete	Jenkins; Gitlab CI
P1	SM-e3	Event	Complete	Jenkins; Gitlab CI
P1	SM-e4	Event	Complete	Jenkins; Gitlab CI
P1	SM-e5	Event	Complete	Jenkins; Gitlab CI
P1	SM-e7	Event	Complete	Jenkins; Gitlab CI
P1	SM-e8	Event	Complete	Jenkins; Gitlab CI
P1	SM-e9	Event	Complete	Jenkins; Gitlab CI
P1	SM-e10	Event	Complete	Jenkins; Gitlab CI
P1	SM-e11	Event	Complete	Jenkins; Gitlab CI
P1	SM-e12	Event	Complete	Jenkins; Gitlab CI
P1	SM-e13	Event	Complete	Jenkins; Gitlab CI
P1	SM-e14	Event	Complete	Jenkins; Gitlab CI
P1	SM-e15	Event	Complete	Jenkins; Gitlab CI
P1	SM-e16	Event	Complete	Jenkins; Gitlab CI
P1	SM-e17	Event	Complete	Jenkins; Gitlab CI

Practice	Id	Type	Automation	Tool
P1	SM-e18	Event	Complete	Jenkins; Gitlab CI
P1	SM-e19	Event	Complete	Jenkins; Gitlab CI
P1	SM-e20	Event	Complete	Jenkins; Gitlab CI
P1	SM-e21	Event	Complete	Jenkins; Gitlab CI
P1	SM-g1	Gate	Complete	Jenkins; Gitlab CI
P1	SM-g2	Gate	Complete	Jenkins; Gitlab CI
P1	SM-g3	Gate	Complete	Jenkins; Gitlab CI
P1	SM-g4	Gate	Transparency	Artifactory
P1	SM-g5	Gate	Complete	Jenkins; Gitlab CI
P1	SM-t1	Task	Partial	RetireJs; OWASP Dependency Check;
P1	SM-t2	Task	Human Task	Cuckoo Sandbox
P1	SM-t3	Task	Partial	OpenSCAP
P1	SM-t4	Task	Human Task	N/A
P1	SM-t5	Task	Partial	Jira
P1	SM-t6	Task	Human Task	N/A
P1	SM-t7	Task	Human Task	N/A
P1	SM-t8	Task	Complete	OpenSCAP; OpenVAS; Prowler; Scout2; vuls
P1	SM-t9	Task	Transparency	Jira

B.2 Practice 2 - Security Requirements Specification

Practice	Id	Type	Automation	Tool
P2	SR-g1	Gate	Human Task	N/A
P2	SR-t1	Task	Partial	Qualys WAS
P2	SR-t2	Task	Human Task	N/A
P2	SR-t3	Task	Partial	OWASP ThreatDragon; Raindance; Microsoft Theat modeling tool
P2	SR-t4	Task	Partial	Microsoft Threat modeling tool
P2	SR-t5	Task	Partial	OWASP ThreatDragon; Raindance; Microsoft Theat modeling tool
P2	SR-t6	Task	Human Task	N/A
P2	SR-t7	Task	Human Task	N/A
P2	SR-t8	Task	Tool Possible	N/A
P2	SR-t9	Task	Tool Possible	N/A
P2	SR-t10	Task	Human Task	N/A

B.3 Practice 3 - Secure Design

Practice	Id	Type	Automation	Tool
P3	SD-e1	Event	Human Task	N/A
P3	SD-g1	Gate	Human Task	N/A
P3	SD-g2	Gate	Human Task	N/A
P3	SD-g3	Gate	Complete	Jenkins; Gitlab CI
P3	SD-g4	Gate	Complete	Jenkins; Gitlab CI
P3	SD-t1	Task	Tool Possible	N/A
P3	SD-t2	Task	Human Task	N/A
P3	SD-t3	Task	Human Task	N/A
P3	SD-t4	Task	Tool Possible	N/A
P3	SD-t5	Task	Complete	Jenkins; Gitlab CI
P3	SD-t6	Task	Complete	Jenkins; Gitlab CI

B.4 Practice 4 - Secure Implementation

Practice	Id	Type	Automation	Tool
P4	SI-e1	Event	Complete	Jenkins; Gitlab CI
P4	SI-g1	Gate	Human Task	N/A
P4	SI-g2	Gate	Transparency	Jira
P4	SI-t1	Task	Human Task	N/A
P4	SI-t2	Task	Human Task	N/A
P4	SI-t3	Task	Partial	**IDE plugins**: DevSkim; FindSecuritybugs; SonarLint **Code Review** : Gerrit **Security coding standards**: OWASP Proactive controls; CERT Secure coding standards **Frameworks:Java**: Spring Security; Shiro; OACC; **Python**: Yosai; Flask Security
P4	SI-t4	Task	Partial	Jira
P4	SI-t5	Task	Partial	OpenSCAP
P4	SI-t6	Task	Complete	Findbugs; 360 Fireline; ESLint; Phan
P4	SI-t7	Task	Human Task	N/A
P4	SI-t8	Task	Partial	Nmap; openVAS; openSCAP

B.5 Practice 5 - Security Verification and Validation Testing

Practice	Id	Type	Automation	Tool
P5	SVV-g1	Gate	Complete	Jenkins; Gitlab CI
P5	SVV-g2	Gate	Human Task	N/A
P5	SVV-t1	Task	Human Task	N/A
P5	SVV-t2	Task	Human Task	N/A
P5	SVV-t3	Task	Human Task	N/A
P5	SVV-t4	Task	Complete	**Unit tests**: JUnit; Mocha; xUnit; **Smoke tests**: ZAP; Baseline scan; nmap **Security Acceptance testing**: BDD--Security; GauntIT; Mittn **Infrastructure Compliance Checks**: inSpec; HubbleStack
P5	SVV-t5	Task	Complete	Blitz; Loader;Blazemaster
P5	SVV-t6	Task	Complete	**Fuzz testing**: Peach; FuzzDB; API-fuzzer
P5	SVV-t7	Task	Partial	Jira
P5	SVV-t8	Task	Partial	CVSS v3 calculator
P5	SVV-t9	Task	Complete	**Fuzz testing**:Peach; FuzzDB; API-fuzzer **Login brute force**: THC Hydra **SQL injection test**: SQLMap, Sqlninja

B.6 Practice 6 - Management of security-related issues

Practice	Id	Type	Automation	Tool
P6	DM-e1	Event	Human Task	N/A
P6	DM-e2	Event	Human Task	N/A
P6	DM-e3	Event	Human Task	N/A
P6	DM-e4	Event	Human Task	N/A
P6	DM-e5	Event	Human Task	N/A
P6	DM-e6	Event	Human Task	N/A
P6	DM-e7	Event	Complete	Jira/ Other project management tool
P6	DM-e8	Event	Complete	Jira
P6	DM-g1	Gate	Complete	Jenkins; Gitlab CI
P6	DM-g2	Gate	Human Task	N/A
P6	DM-g3	Gate	Human Task	N/A
P6	DM-g4	Gate	Transparency	Jira
P6	DM-g5	Gate	Transparency	Jira

Practice	Id	Type	Automation	Tool
P6	DM-t1	Task	Human Task	N/A
P6	DM-t2	Task	Complete	Jira
P6	DM-t3	Task	Transparency	Jira
P6	DM-t4	Task	Human Task	N/A
P6	DM-t5	Task	Transparency	Jira
P6	DM-t6	Task	Transparency	Jira
P6	DM-t7	Task	Human Task	N/A
P6	DM-t8	Task	Partial	CVSS v3 calculator
P6	DM-t9	Task	Human Task	N/A
P6	DM-t10	Task	Human Task	N/A
P6	DM-t11	Task	Partial	Jira
P6	DM-t12	Task	Human Task	N/A

B.7 Practice 7 - Security Update Management

Practice	Id	Type	Automation	Tool
P7	SUM-e1	Event	Tool Possible	N/A
P7	SUM-e2	Event	Tool Possible	N/A
P7	SUM-g1	Gate	Complete	Jenkins; Gitlab CI
P7	SUM-t1	Task	Tool Possible	N/A
P7	SUM-t2	Task	Partial	OpenSCAP; OpenVAS; Prowler; Scout2; vuls
P7	SUM-t3	Task	Partial	TestingWhiz; SahiPro; TestComplete; Silk Test; IBM Rational Functional Tester
P7	SUM-t4	Task	Human Task	N/A
P7	SUM-t5	Task	Complete	Jenkins; Gitlab CI
P7	SUM-t6	Task	Tool Possible	N/A
P7	SUM-t7	Task	Complete	Jenkins; Gitlab CI
P7	SUM-t8	Task	Complete	

B.8 Practice 8 - Security Guidelines

Practice	Id	Type	Automation	Tool
P8	SG-t1	Task	Human Task	N/A
P8	SG-t2	Task	Human Task	N/A
P8	SG-t3	Task	Human Task	N/A
P8	SG-t4	Task	Human Task	N/A
P8	SG-t5	Task	Human Task	N/A
P8	SG-t6	Task	Human Task	N/A
P8	SG-t7	Task	Human Task	N/A
P8	SG-t8	Task	Human Task	N/A
P8	SG-t9	Task	Human Task	N/A
P8	SG-t10	Task	Transparency	N/A

Appendix C

Questionnaire

This part of the appendix presents the questionnaire used in the evaluation phase of the proposed solution as mentioned in Chapter 5.

Questionnaire

Dear participant:

This questionnaire takes part in the evaluation phase of a master thesis with the aim of evaluating the developed methodology and its suitability for specifying a DevOps pipeline compliant to the IEC 62443-4-1 (4-1) standard.

The questionnaire is voluntary and anonymous, however your role at the company is the only personal information you need to fill in for statistical and evaluation purposes. Your answers are relevant for the study regardless of your role at the company or your level of expertise in the topic.

The questionnaire contains open-ended questions and statements about the familiarity of the users related to specific concepts. Please indicate the extent to which you are familiar or not with each of the presented concepts and tools' usability, making use of the scale provided in each case. Here is an example:

		Beginner	Medium	Advanced	Expert	No answer
ID	Statement	1	2	3	4	
I.1	Are you familiar with the stated concept?					

Mark the following column to indicate the degree to which you are familiar with the mentioned concept in the statement, from "**Beginner** to "**Expert**" with an 'X'. If for some reason you cannot or do not wish to reply, you should mark the last column "**No answer**" with an 'X'.

Thank you very much for your cooperation.

Role at Company: _____

Introductory Questions

		Beginner	Medium	Advanced	Expert	No answer
ID	Statement	1	2	3	4	
I.1	Are you familiar with the 4-1 standard?					
I.2	Are you familiar with DevOps?					
I.3	Are you familiar with DevSecOps?					
I.4	Are you familiar with security tools?					

Part 1 - Are the process models and tooling information suitable to specify a DevSecOps pipeline compliant to 4-1?

Question 1.1 - Is this classification criteria precise?

Question 1.2 - Is this classification criteria helpful for building a pipeline?

Part 2 - Is the suggestion for automation of 4-1 in the pipeline feasible?

Question 2.1 - Do you agree that the selected tools fully automates the corresponding tasks?

Question 2.2 - Do you agree that the selected tools partially automates the corresponding tasks?

Question 2.3 - Do you agree that the selected tools aids a practitioner with a visualisation of a problem?

Question 2.4 - Do you agree that the selected tasks cannot be automated?

Question 2.5 - Do you agree that the selected tasks are straightforward enough for a tool to be built?

Question 2.6 - Have you ever seen any of these tools before?

Question 2.7 - Do you suggest any other tool to fulfil the observed tasks?

Part 3 - Do you think the results shown in the graphics are relevant?

Question 3.1 - Do the observed graphics correspond to your expectation of possible 4-1 automation?

Question 3.2 - Are these insights helpful for your daily work?

Question 3.3 - Which benefits and drawbacks do you envision from this approach?

Bibliography

Alcaraz, C., & Zeadally, S. (2015). Critical infrastructure protection: Requirements and challenges for the 21st century. *International journal of critical infrastructure protection*, *8*, 53–66.

Atlassian. (2016). *Devops: Breaking the development-operations barrier.* Retrieved from `https://www.atlassian.com/devops` (Accessed: 08.09.2019)

Beck, K., Beedle, M., Van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., ... others (2001). Manifesto for agile software development.

Ben Othmane, L., Jaatun, M. G., & Weippl, E. (2017). *Empirical research for software security: Foundations and experience.* CRC Press.

Bird, J. (2016). Security as code: Security tools and practices in continuous delivery. In *Devopssec: Securing software through continuous delivery* (p. 32-36). O'Reilly Media, Incorporated.

Blumano. (2018). *Iec 62443: Levels, levels and more levels.* Retrieved from `https://blumano.com/iec-62443-levels-levels-and-more-levels/` (Accessed: 17.09.2019)

Dännart, S., Constante, F. M., & Beckers, K. (2019). An assessment model for continuous security compliance in large scale agile environments. In *International conference on advanced information systems engineering* (pp. 529–544).

DORA. (2018). *Accelerate: State of devops. strategies for a new economy.* DevOps Research Assessment. Retrieved from `https://cloudplatformonline.com/rs/248-TPC-286/images/DORA-State%20of%20DevOps.pdf` (Accessed: 31.05.2019)

Fitzgerald, B., & Stol, K.-J. (2014). Continuous software engineering and beyond: trends and challenges. In *Proceedings of the 1st international workshop on rapid continuous software engineering* (pp. 1–9).

Gartner. (2017). *10 things to get right for successful devsecops.* Retrieved from `https://emtemp.gcom.cloud/ngw/eventassets/en/conferences/lsc37/documents/gartner-io-cloud-us-research-note-successful-devsecops-2018.pdf` (Accessed: 03.06.2019)

Hevner, A. V., March, S. T., Park, J., & Ram, S. (2004). Design science in information systems research. *MIS quarterly*, *28*(1), 75–105.

Highsmith, J., & Cockburn, A. (2001). Agile software development: The business of innovation. *Computer*, *34*(9), 120–127.

Hsu, T. H.-C. (2018). *Hands-on security in devops: Ensure continuous security, deployment, and delivery with devsecops.* Packt Publishing Ltd.

Humble, J., & Farley, D. (2010). *Continuous delivery: Reliable software releases through build, test, and deployment automation.* Pearson Education.

IEC. (2014). Iec 62443-1-1 security for industrial and automation control systems part 1-1: Models and concepts. *International Electrotechnical Commission*.

IEC. (2018). Iec 62443-4-1: 2018: Security for industrial automation and control systems-part 4-1: Secure product development lifecycle requirements. *International Electrotechnical Commission*.

IEC. (2019). Iec 62443-4-2: 2019: Security for industrial automation and control systems-part 4-2: Technical security requirements for iacs components. *International Electrotechnical Commission*.

ISA. (2017). *Isa-99/iec 62443, industrial automation and control systems security.* International Society of Automation. Retrieved from `https://isaeurope.com/iec62443/` (Accessed: 29.08.2019)

ISA. (2018a). *Isa99, industrial automation and control systems security.* International Society of Automation. Retrieved from `https://www.isa.org/isa99/` (Accessed: 05.06.2019)

ISA. (2018b). *Isa/iec 62443 standard specifies security capabilities for control system components.* Retrieved from `https://www.isa.org/intech/`

201810standards/ (Accessed: 14.05.2019)

ISCI. (2018). Sdla-300, security development life-cycle assurance - certification requirement, version 1, revision 3. *ISA Security Compliance Institute*.

ISO. (2017). *International organization for standardization - benefits of standards.* Retrieved from https://www.iso.org/benefits-of-standards.html (Accessed: 16.05.2019)

LEI. (2009). *Lean enterprise institute - what is lean?* Retrieved from https://www.lean.org/WhatsLean/ (Accessed: 19.07.2019)

Lieberman, H., Paternò, F., Klann, M., & Wulf, V. (2006). End-user development: An emerging paradigm. In *End user development* (pp. 1–8). Springer.

Lwakatare, L. E., Kuvaja, P., & Oivo, M. (2015). Dimensions of devops. In *International conference on agile software development* (pp. 212–217).

Mohan, V., & Othmane, L. B. (2016). Secdevops: Is it a marketing buzzword?-mapping research on security in devops. In *2016 11th international conference on availability, reliability and security (ares)* (pp. 542–547).

Moyon, F., Beckers, K., Klepper, S., Lachberger, P., & Bruegge, B. (2018). Towards continuous security compliance in agile software development at scale. In *2018 ieee/acm 4th international workshop on rapid continuous software engineering (rcose)* (pp. 31–34).

Nair, S., de la Vara, J. L., Sabetzadeh, M., & Falessi, D. (2015). Evidence management for compliance of critical systems with safety standards: A survey on the state of practice. *Information and Software Technology*, *60*, 1–15.

Puppet. (2018). *State of devops report, 2018.* Retrieved from https://www.thinkahead.com/wp-content/uploads/2018/10/State-of-DevOps-Report.pdf (Accessed: 03.06.2019)

Rahman, A. A. U., & Williams, L. (2016). Software security in devops: synthesizing practitioners' perceptions and practices. In *2016 ieee/acm international workshop on continuous software evolution and delivery (csed)* (pp. 70–76).

SANS. (2018). *Sans secure devops tooolchain and securing web application technologies checklist.* Retrieved from https://www.sans.org/

security-resources/posters/appsec/secure-devops-toolchain-swat-checklist-60 (Accessed: 12.07.2019)

Scaled Agile, I. (2017). *Safe: What is safe?* Retrieved from https://www.scaledagile.com/enterprise-solutions/what-is-safe/ (Accessed: 03.09.2019)

Semaphore. (2019). *Ci/cd pipeline: A gentle introduction.* Retrieved from https://semaphoreci.com/blog/cicd-pipeline (Accessed: 29.08.2019)

Shahin, M., Babar, M. A., & Zhu, L. (2017). Continuous integration, delivery and deployment: a systematic review on approaches, tools, challenges and practices. *IEEE Access*, *5*, 3909–3943.

Shull, F., Singer, J., & Sjøberg, D. I. (2007). *Guide to advanced empirical software engineering.* Springer.

Sonatype. (2019). *Devsecops community survey, 2019.*

Synotive. (2017). *What are the life cycle models of software development?* Retrieved from https://www.synotive.com/blog/software-development-client-questionnaire (Accessed: 10.09.2019)

Techopedia. (2011). *What is software development life cycle?* Retrieved from https://www.techopedia.com/definition/22193/software-development-life-cycle-sdlc (Accessed: 30.09.2019)

VersionOne. (2011). *6th annual state of agile survey: The state of agile development,".* Retrieved from https://www.versionone.com/pdf/2011-state-of-agile-survey.pdf (Accessed: 14.05.2019)

Wang, X., Conboy, K., & Cawley, O. (2012). "leagile" software development: An experience report analysis of the application of lean approaches in agile software development. *Journal of Systems and Software*, *85*(6), 1287–1299.