

A Work Project, presented as part of the requirements for the Award of a Master's degree in
Management from the Nova School of Business and Economics.

HOW TO COMPARE DEVOPS, DEVSECOPS, AND SCRUM USING ENTERPRISE
ARCHITECTURE MODELING

JAD BCHELLY - 53379

Work project carried out under the supervision of:

Professor Paulo Faroleiro

18/12/2023

Abstract

We live in a century where publishing systems and applications have become a norm, where hundreds of thousands of them are produced on a daily basis. Choosing an organisation with a fast product development lifecycle, adapting to market changes, and complying with governmental laws is a must to keep companies competitive in this growing market. To better understand the differences and similarities between different methodologies and frameworks used by development companies, this paper compares DevOps, DevSecOps, and Scrum using the Enterprise Architecture (EA) approach using the ArchiMate modelling language.

Keywords: DevOps, DevSecOps, Scrum, Enterprise architecture, modelling, product development life cycle

Table of Contents

| | |
|---|-----------|
| Abstract | 1 |
| Introduction | 3 |
| Literature review | 4 |
| Literature review on Enterprise Architecture | 4 |
| Definition of Enterprise Architecture | 4 |
| Benefit of integrating Enterprise Architecture | 4 |
| Literature review on pre-DevOps | 5 |
| Literature review on DevOps | 6 |
| The emergence of DevOps | 6 |
| DevOps phases | 7 |
| Literature review on DevSecOps | 10 |
| The emergence of DevSecOps | 10 |
| DevSecOps phases | 10 |
| Literature review on Scrum | 12 |
| Scrum definition | 12 |
| Scrum process and Scrum Sprint process | 13 |
| Sprint planning | 14 |
| Sprint | 14 |
| Research design | 16 |
| Problem | 16 |
| Research question | 17 |
| Proposal | 17 |
| Results | 18 |
| DevOps | 18 |
| DevSecOps | 19 |
| Scrum | 20 |
| DevOps, DevSecOps, Scrum | 21 |
| Conclusion | 24 |
| Directions for future research | 25 |
| References | 26 |
| Appendix | 28 |

Introduction

Being up-to-date, relevant, and compliant is what makes an organisation successful. (BEA Oghojafor, OO Olayemi, OO Oluwatula, 2012). Nowadays, people lean towards using reliable, functional systems and applications that bring value to their daily lives. Organisations are looking for well-grounded software development companies to take charge of building what is needed quickly, authentically, and in a stable way. That is where experienced development companies shine in determining the appropriate framework or methodology based on the complexity of the requirements, the client's expectations, and the project's timeline. (HH Olsson, J Bosch, H Alahyari, 2013).

This paper focuses on three famous frameworks and methodologies software development companies use: DevOps, DevSecOps, and Scrum. This paper aims to answer the question: "**How to compare DevOps, DevSecOps, and Scrum using Enterprise Architecture modeling?** "

No previous attempts have been made available that include using EA modelling to compare them. More specifically, using ArchiMate, an open-source, independent modelling language for EA that helps describe, study, and visualise the interconnection among architecture domains in an evident way. (The Open Group, 2016).

To help answer these questions and reach proper conclusions, the following steps were conducted. Starting with a "**Literature review**" provides a theoretical context for the research and hypotheses. "**Research Design**" is then brought up by highlighting the resources and strategies used that are suitable for the research. "**Results**" later demonstrate the research outcome and reveal the findings deduced. Finally, the "**Conclusion**" summarises the entire research and highlights discoveries. The identified limitations are also acknowledged and, in conclusion, potential directions for further research on these topics is suggested.

Literature review

Literature review on Enterprise Architecture

Definition of Enterprise Architecture

Enterprise Architecture is a comprehensive framework applied by businesses from different industries to help them develop a structured approach to designing, planning, managing, and implementing their resources to support their objectives and business goals (Gama, 2013).

In fact, there is no official definition for EA. Since the 1960s, many perspectives have come to light on what EA is, influenced by the type of industry, culture, organisational structure, and diverse stakeholder backgrounds the organisations have. It is also important to mention that EA is not a static framework that every organisation should abide by. It evolves with technological advancements and new methodologies, leading to multiple interpretations, each reflecting a different course and alteration.

However, one sentence written by The Open Group in 2018 clarifies the purpose of implementing the EA framework at the heart of any organisation's core strategy: "Enterprise Architecture is the process of translating business vision and strategy into effective enterprise change by creating, communicating, and improving key requirements, principles, and models that describe the enterprise's future state and enable its evolution." (The Open Group, 2018).

Benefits of Integrating Enterprise Architecture

In their research paper titled "Enterprise Architecture: Enabling Integration, Agility, and Change", theorists and researchers Jeanne W. Ross, Peter Weill, and David C. Robertson stated, in 2006, that "Three core imperatives are essential for modern businesses and organisations: seamless integration of customer and operational processes, agility, and the ability to change". However, they point out that these imperatives are not satisfied, and Enterprise Architecture should be

integrated to link strategy development and execution. They emphasise the design and structure of systems rather than just their functional aspect and the execution part. In fact, integrating EA ensures that the organisation's systems align with the company's goals and objectives. It also supports agility and brings along the capacity to change and adapt to changing environments and shifts in business needs. In addition, EA facilitates communication between different stakeholders involved in developing a project from its strategy development to its execution by helping them understand how strategic decisions play an essential role in the overall organisational structure. Moreover, since it allows an understanding of the organisation's architecture, EA helps anticipate potential challenges, preparing the team to reduce them cautiously (Bredmeyer, E, 2004).

Literature review on pre-DevOps

In their Phoenix project book, written by Gene Kim, Kevin Behr, and George Spafford (2013), the authors mention how challenging it was to develop a system without collaboration and visibility. To build a system for its clients, a development company needed two separate teams: The development team and the operations team. Each one of them works independently and has a different working agenda.

The development team's primary goal is to create and enhance the application's functionality to meet business requirements. The team is responsible for developing the code and integrating the system's functionalities and features. Their role is also to implement new features, modify functionalities, optimise code, and fix bugs (Mainak, 2019).

The main focus of the operations team is to ensure the availability, performance, and stability of the application in the product environment, which translates to the following: Deploying the application on servers, configuring the necessary infrastructure, and setting up databases. This team plays a significant role in identifying issues derived from the application production phase, and it's

their responsibility to troubleshoot and resolve them. During the whole process, from starting to work on the project to the delivery phase, no interaction is made between the two departments as they have distinct roles and responsibilities (Dobra, 2021). It is essential to remember that the operational team will only start working once it receives the application from the development team. Back in the days, the typical application or system would take months or even years to be fully developed and operational. Some of the reasons that led to the slow pace of building applications and systems were the manual configuration and deployment (lack of technological automation), lack of collaboration and visibility between teams (lack of product lifecycle strategy), scalability challenges as well as maintenance disruption (Haththakage, 2023).

Literature review on DevOps

The emergence of DevOps

Year after year, project after project, the complexity of systems and applications increased, and it became clear that traditional development and deployment methods struggled to keep pace with business demand. It became comprehensible that something needed to change, and closer collaboration between development and operations teams was demanded. (Mojtaba Shahin, and M. Ali Babar, 2020). It wasn't official until 2009, when a new chapter started for the world of technology and a conference titled "DevOpsDays" took place in Belgium (Modi, 2023).

As Amazon Web Services (AWS, 2022) defines it: "DevOps is the combination of cultural philosophies, practices, and tools that increases an organisation's ability to deliver applications and services at high velocity: evolving and improving products at a faster pace than organisations using traditional software development and infrastructure management processes." The company has also provided a model showing how they operate together (Appendix 1). In other words, it is better

practice and more efficient to keep the development and the IT operations teams more involved with each other to deliver better and faster performance up to the quality demanded from them.

DevOps phases

The DevOps culture is implemented in multiple phases with the help of several tools. (Manish Virmani, 2015). It all starts with the **planning phase**, where the product owner or manager puts down a plan keeping in mind application objectives and requirements needed to be delivered to the client. A Business Analyst meets with the stakeholders (clients and involved actors) to better understand what is required from the team to achieve success at the end of the product lifecycle. Once the plan is set, it is essential that the Development Engineers and CI/CD Engineers set up and manage the version control system correctly. In more detail, version control is an essential component of DevOps because it guarantees that any modification made to the program codebase is monitored, recorded, and available to the whole development team. That happens when the engineers merge and push their code changes to a shared repository accessible to the whole team (Atlassian, 2023).

After the Development Engineers' tasks are fulfilled, the second phase starts and the development team is set to start **coding**. The team work on different code versions while saving them in a repository. To have a fast development lifecycle, meet the deadline, and work efficiently, multiple Developers work at the same time. Each developer has their own set of tasks to take care of.

Once the code is written, the **build** phase is considered to be reached. What happens next is the **testing** phase. In the testing phase, a quality assurance team is tasked with ensuring that the code developed by the development team meets established quality standards. The code quality is verified by writing and executing proper code tests, including unit and integration tests. A crucial

part of this phase that brings value to the DevOps methodology is the use of continuous integration (CI) and continuous development (CD) (Jakob Pennington, 2019).

The same platform where the repository is present is used for continuous integration. As described by IBM: "Continuous integration is a software development process where Developers integrate the new code they've written more frequently throughout the development cycle. Automated testing is done against each iteration of the build to identify integration issues earlier, when they are easier to fix, which also helps avoid problems at the final merge for the release." (IBM, 2022).

Continuous Development enhances Continuous Integration by frequently integrating code changes into a shared repository, where automated builds and tests are triggered to ensure that new code changes do not break the existing functionality (Christopher Cowell, Nicholas Lotz, and Chris Timberlake, 2023).

After testing, the work transitions from the development team to the operations team to manage the remaining phases of the DevOps methodology. The **deployment** phase begins. A deployment phase is when the code previously established in version control by Developers has been retrieved and made readily available to the end user. Deployment Engineers use scripts and automation tools to automate the deployment process. This covers operations like file copies, configuration updates, and service restarts. They oversee and set up the target environments, ensuring they meet the program's requirements to function correctly (Surabattina Sai Sravan, 2023). During the deployment, the operation team stays on high alert to respond to any issues that might arise and ensure that rollout procedures are in place to return to the previous stable version in case the deployment encounters critical issues due to a minor error or change. Release managers play an essential role by ensuring that software updates are distributed smoothly and under control. After that, some post-deployment measures are implemented to monitor the application's performance,

manage scaling and handle unexpected errors promptly. That is why **Operate, Monitoring and Feedback** come next. Professionals, including Monitoring and Development Engineers, are involved in this stage to guarantee peak performance, keep the system online, and quickly detect any potential problems. They put monitoring and alerting systems specifically built to track the application's performance, detect anomalies and send alerts if a pre-defined performance and application behaviour deviates. Once anomalies are detected, they work towards fixing them immediately and efficiently (Candy Pang, 2016).

By taking advantage of the technologies created and components employed in the process, the goals of DevOps were achieved. The first one is to shorten the software development lifecycle (SDLC). In other words, this goal aims to minimise as much as possible the time it takes to move from the idea to the delivery of working software — it is achieved by using automation, CI and CD. The second one is increasing the frequency of software releases. This success is mainly generated because of the automation of deployment. The third is improving the Reliability of Software releases. Reliability is essential because when a software or application is reliable, it helps avoid outages. Reliability is established once monitoring is in place (Akshit Raj, Sulabh Tyagi, 2022).

In summary, under a DevOps model, development and operations teams are no longer isolated from each other. These two teams collaborate, where the engineers work across the whole application lifecycle — from development and testing to deployment and operations. This new way of working helped address issues earlier, sped up deployment cycles, and encouraged a culture of shared responsibility towards the application being built.

Literature review on DevSecOps

The emergence of DevSecOps

In pre-DevOps, the development team and the operations team worked independently from each other. After years of practice and trying to find better ways of operating, teams realised that the development and operations teams could form a collaboration that could lead to a faster System Development Life Cycle (SDLC). After cybersecurity threats emerged and regulations put pressure on organisations from GDPR and HIPAA that imposed strict data protection and privacy standards, DevSecOps was introduced as a solution to fix all these problems (Neil Macdonald, and Ian Head, 2016).

As IBM defines it: "DevSecOps - short for development, security, and operation - integrates application and infrastructure security seamlessly into Agile and DevOps processes and tools. It addresses security issues as they emerge when they're easier, faster, and less expensive to fix. (IBM, 2022). In other words, DevSecOps is when all three teams collaborate to prioritise security without slowing down the development process. It also ensures that security is everyone's concern during the product development lifecycle.

DevSecOps phases

No significant difference exists between how DevOps and DevSecOps execute and implement its strategies. They follow almost the same agenda from the planning phase to the deployment and product release phase, adding additional security measures. During the **planning** phase, teams conduct security analysis and develop an agenda for security testing that specifies when, how, and in which components it will take place. Security comes before coding, enabling Developers to identify and address potential vulnerabilities as they write code. Doing so will reduce the risk of unintentionally leaving vulnerabilities and holes in the final product. In the **coding** phase, every

commit (code submitted) and merge will automatically trigger a security test or review after security technologies are directly integrated into the developer's repository workflow set up by a security engineer. Some popular security tools used are CheckStyle, PMD, and SpotBugs. Once the repository is filled with code, the **build** phase takes place where automated security analysis tools such as Checkmarx, Snyk, and OWASP Dependency-Check scan the code for potential security flaws (Agung Maulana, Herman Kabetta, 2022). It is a critical procedure because it minimises human error and is time-efficient as it scans the code quickly, looking for mistakes. After a build artifact has been successfully created and sent to staging or testing environments, the **test** phase begins. It takes a lot of time to run through an entire test suite (various individual tests targeting different functionalities).

To reserve more resource-intensive and timely tests for later and optimise efficiency, the testing stage needs to identify errors quickly using fast testing methods. Throughout the testing process, dynamic application security testing (DAST) tools identify application processes, including authorisation, user authentication, endpoints related to APIs, and SQL injection (Havard M and Ricardo C, 2017). This method guarantees that resources are allocated wisely and that significant problems are found early in the testing process. By the time the DevSecOps cycle reaches the **release** phase, the application should have gone through extensive testing. This stage reviews environment configuration values, including network firewall, personal data management, and user access control (Tao Chen, and Haiyan Suo, 2022). The **Deployment** phase comes next, where the team should carefully examine configuration disparities between the current production environment and the initial development setting.

A short example is supposing the application database connection settings were configured for testing purposes without considering high-security measures. During the deployment, the security

team ensures that the database's configuration is set at the highest level of security. That minimises the risk of database attacks that can leak user (client) sensitive data (Mary S, Ricardo C, 2020). Lastly comes **Operation and Monitoring**. This last phase is the cherry on top regarding ensuring that security measures are always ready to keep the product safe. Periodic maintenance of Zero-day vulnerabilities is crucial to track and resolve them as quickly as possible. Vulnerabilities discovered should be addressed instantly to avoid attackers and black hat hackers from leveraging it and attacking the overall infrastructure. The security team can use Infrastructure as Code (IaC) to help them manage and provision infrastructure resources, making it easier to keep code versions and replicate the infrastructure environment (Nenad Petrovic, and Matija Cankar, 2022). A continuous monitoring tool can be put in place to keep track of the system's performance in real-time that helps spot any exploits at an early stage so the security team, with the collaboration of the Developers, can attack and fix the vulnerabilities before they spread in the wrong person's hands (Vaishnavi Mohan, and Lotfi Ben Othamane, 2016).

Overall, introducing DevSecOps as a security layer to DevOps helped Increase security, speed and efficiency, created cost savings, and enforced continuous improvement. As technology evolves, intruders are always looking for new processes and techniques to hack systems, which they mostly get their hands on. DevSecOps integrates security during the lifetime cycle of a program by ensuring that all measures and new technology updates and processes are considered, which helps programs stay away from emerging hacking tools and reduce their risk of being compromised.

Literature review on Scrum

Scrum definition

As defined by AWS: "Scrum is a management framework that teams use to self-organise and work towards a common goal. It describes a set of meetings, tools, and roles for efficient project delivery" (AWS, 2022).

The primary Scrum team comprises a Product Owner, a Scrum Master, and several Developers. In a nutshell, each one of them has the following responsibilities (Kittitouch Suteeca; Sakgasit Ramingwong, 2016)

The product owner represents the stakeholder, i.e. the customer or the end user. The Scrum Master ensures that the scrum process runs as smoothly as possible, helps the team remove and deal with obstacles, and ensures that teams and processes adhere to the Scrum principles. The development team is responsible for building the product overall. They get their hands dirty in building the applications (coding) and respond to all requirements and functions to ensure a successful final product delivery (Ken Schwaber, 2019).

Scrum process and Scrum Sprint process

The scrum process is composed of multiple sprints, each with a specific goal to reach. Each sprint takes about one to three weeks to be completed. The main goal of multiple sprints is to produce a valuable product increment, adapt to feedback, and improve to get to a complete and valuable product. The more sprints we have, the more product precision and the fewer uncertainties appear. However, the number of sprints a team completes depends on the project's scope and complexities (Christoph Matthies, 2019).

The Scrum process always starts with creating a product backlog initiated by the product owner. The product backlog, agile, acts as a dynamic and ongoing document that collects and arranges all the features, user stories, improvements, problem patches, and other things that need to be addressed in a product. The product backlog goes through ongoing updates based on emerging requirements.

Sprint planning

To keep it organised, the Scrum team holds a Sprint planning meeting to define the work for the upcoming sprint. During the meeting, the product owner lists the highest priority items from the product backlog that they wish to see finished during the following sprint. The development team ensures they understand these items and the work required to achieve them. The Scrum Master's job is to facilitate the meeting, ensuring that the Scrum framework is followed and adapted. In addition, the team examined the technicalities and approaches for completing selected tasks. A sprint backlog is created once the sprint planning meeting is successful and decisions are drawn (Victor Faniran, 2017).

Sprint

The most essential part of the Sprint process is the **Daily Scrums**, also called Daily Standups. Daily Scrums is a meeting held by the Scrum Master and the Developers every day during the sprint. During this time-boxed meeting, which usually takes approximately 10 to 15 minutes, team members discuss and share what they worked on during the sprint (the previous day) and what they plan to work on during the next one (Rashmi Popli, and Naresh Chau, 2013).

After the daily scrum, the development team takes action and works on **developing** the product and implementing the tasks they aligned on. Their main tasks include designing the product, writing the code to execute, and ensuring that everything they implement is secure and tested. Developers work simultaneously by implementing their code to a version control repository, which helps manage and track ongoing changes to the code (Jessica R, and Frand F, 2009).

In the Scrum framework, there is no particular and separate security team. Instead, security responsibility is distributed across the entire Scrum team. **Security** practices are incorporated into the entire software development lifecycle. Some security practices include security code reviews

and integrating automation practices to secure the code implemented during the development process. They help mitigate security vulnerabilities at earlier stages, give room for improvement, and tackle code weaknesses (Imran Ghani, 2011).

After each sprint, the product incrementally takes shape. Every sprint plays a crucial role in the scrum process. The team continuously looks for ways of product improvement and work based on feedback and evolving requirements from the sprint beforehand. At the end of each sprint, the product should be in a shippable state, meaning it works properly as previously planned and is free of flaws, which is called being at an **Incremental state**.

Following that comes the most critical part: ensuring everything was well done. During this phase, the Scrum team hold a **Sprint Review** meeting where the development team demonstrate the product they built during the sprint to the stakeholders. After discussions and conclusions, the stakeholders provide feedback on the overall sprint process and can suggest changes or draw new requirements that the development team will take care of during the upcoming sprint (Jeff Sutherland, 2004).

After the Sprint review, a **Sprint Retrospective** meeting takes place to reflect on the sprint's successes and areas of improvement to enhance efficiency and productivity during the succeeding sprint. The actors involved are the Scrum Master, the Product Owner, and the Development team.

After the feedback and insights drawn from the Sprint review and the Sprint retrospective meetings, the team make all necessary adjustments to the product backlog. This process is called "**Backlog refinement**". The team make sure that the problems do not happen again. Once the Sprint review, sprint retrospective, and backlog refinement are concluded, the sprint backlog gets adjusted and ready for the sprint to come (Scrum Inc., 2012).

After all sprints take place, considering all adjusted backlogs; and the product reaches its final incremental stage, a **final sprint review** is conducted, where the product gets demonstrated in front of stakeholders who provide feedback for final adjustments. After that, a **product release phase** occurs, consisting of the product delivered to the client. There is no operations team to take care of the product release and make it available to the end user. Instead, the Product Owner is mainly involved with the product release amongst the Scrum Team.

An essential part after the product release is feedback and monitoring. Some organisations assign this role to a Monitoring engineer; however, it is also the role of the whole Scrum team to ensure that performance monitoring is taken care of. Alerting tools are set up to immediately trigger warnings when anomalies occur to chase and fix system issues before users get affected (Fabio Rocha, Guillermo Rodriguez, 2022).

Scrum has shown to be an agile methodology that places a strong emphasis on teamwork, flexibility, and incremental advancement.

Research design

Problem

As mentioned in the literature review, DevOps, DevSecOps, and Scrum are practices that have evolved over the years based on project requirements, feedback, market demand, and shifts. Depending on the project being worked on, considering its complexity, features required, and timeline, software development companies choose the framework and methodology that fits the best to deliver a functional and impeccable project for their clients.

In this matter, EA can be a modelling tool to help visualise, analyse, and compare different ideologies and frameworks within their overall structure and objectives.

Several pieces of literature and publications cover the ideology behind the benefits and limitations of using DevOps, DevSecOps, and Scrum. However, none of them draws a comparison or finds similarities between them using EA modelling.

Research question

Considering the information presented earlier, the paper explored the subsequent research query:

Q: How can enterprise architecture modelling be used to compare DevOps, DevSecOps, and Scrum to find their similarities and differences?

Proposal

A qualitative and empirical approach was used to provide a comprehensive and insightful investigation and address the question. First, the research aimed to identify key reasons software development companies use DevOps, DevSecOps, and Scrum and understand how and what led them to evolve in the market. Second, it is crucial to understand the workflow and the role of the actors of the methodologies and frameworks in question. Third, to see the similarities and differences between DevOps, DevSecOps, and Scrum and perceive how the workflow, roles, and actors are interconnected in these approaches, the ArchiMate modelling language was applied based on the TOGAF standard, the most renowned methodology for Enterprise Architecture (The Open Group, 2018). The research sources were online libraries, including Google Scholar, Institute of Electrical and Electronics Engineers (IEEE), JSTOR, and additional books and blog content.

Three models have been developed and decomposed into elements and relationships using Archi's modelling toolkit. The first model represents DevOps. The second model represents DevSecOps. The third and final model represents Scrum. Finally, the models were analysed and compared to understand their similarities and differences.

Results

For the sake of brevity, Archi representations are explained in Appendix 2.

DevOps

As previously mentioned in the Literature Review, DevOps is a methodology in the software development industry. A modeling of DevOps is presented in Appendix 3. After the development company assigns the project to be developed using a DevOps methodology, Business Actors, such as the Product Owner and the Business Analyst, meet (called Business process in EA) to define the product backlog, represented as a business object. Development Engineers and Infrastructure Specialists then collaborate to set up the infrastructure that plays an essential role in the next stage: development and deployment. During this stage, Development Engineers and Software Developers have a common assignment: to develop the code (Application process) applying CI, representing a business process as it builds upon defining a set of product and tactical demands. The CD is represented as a principle and is associated with CI, as it triggers tests whenever code is merged into the repository. CI/CD Engineers have association and assignment relation towards CI/CD as they ensure they operate effectively. Quality Assurance Engineers are responsible for ensuring code quality and setting up security testing, both technology events, because they might change the state of the code developed. Actors such as Security Specialists also play a role in arranging security tests. CI, code quality, and security testing have a flow relation towards the code as they play a role in information transfer to make it more robust.

After the code is finalised and well-secured, Deployment Engineers, with the help of Development Engineers, work on deploying the code and delivering it to the client. Deployment Engineers have a serving relation towards the final product as they provide its functionality by making it accessible to the client, and the Development Engineers have a flow relation towards the final product as they

fix issues that arise and assist Deployment Engineers in making decisions. The last role remains for the Monitoring Engineers, who have a flow relation towards the final product because they set up systems that communicate information about the continuous health of the product and ensure it runs as planned.

DevSecOps

Based on the literature review, it is clear that DevSecOps follows the same layers as DevOps but implements more security layers during the development lifecycle, as presented in Appendix 4. During the project planning, a Security Analyst joins the meeting with the Business Analyst and Product Owner and has an assignment relationship responsible for building a security agenda that will be executed during the product development. The infrastructure setup remains the same and is handled by the development engineer and Infrastructure Specialist. Before starting to code, a security engineer is assigned a business role and has a flow relation with the Software Developers and Development Engineers because he advises them in advance of potential security obstacles they might need to focus on carefully to minimise the risk of possible vulnerabilities. Security Engineers have an assignment relation with CI because they built in a security test that gets triggered whenever a code gets pushed into the version control. That makes the code more secure and minimises human errors. Once the code has been finalised, QA Engineers ensure that the quality of the code is well-formatted, free of errors and ready to be released. Security Specialists run tests to ensure the code is safe to release.

Once the code is secured and finalised, Security Engineers ensure the system's settings are configured and ready to be deployed to prevent configuration mistakes. The configuration is considered a technology event as it makes necessary updates to the system to make it ready for delivery. Deployment Engineers then deploy the code, which is later monitored by Monitoring

Engineers and fixed by the development engineer in case of any trouble. In addition to Security Engineers making sure the configurations are correct, they also take care of the maintenance of the system after being deployed to ensure that vulnerabilities don't arise with the advancement of technologies that enable code penetration.

Scrum

As defined in the literature review, Scrum is an agile framework that helps teams stay organised to achieve efficient project delivery. Its model is presented in Appendix 5. Starting with Sprint Planning, the Scrum Master has a business role because of his specific responsibilities, meeting with the business actors, the Product Owner, and the Developers to create the sprint backlog. It's an assignment related to the three of them, as they are responsible for its creation and execution. After that, the Scrum Master and Developers meet during the daily scrum to align what to work on during the upcoming sprint. Meetings are usually business interactions because multiple actors collaborate to perform a task. Next, the Developers have an assignment relationship because they handle the coding and testing being accomplished during a sprint. A flow relation comes between the daily scrum meeting and the development of the day because Developers transfer what they agreed on during the meeting into a functional code. The security part of the code is represented as a technology event because the code gets updated if security issues are identified. After a sprint, the product reaches an incremental development stage, represented as a product because it combines multiple elements (secured code).

Following the incremental stage, three types of meetings take place. The product is demonstrated before the stakeholders in a sprint review meeting. A sprint retrospective meeting is where product strengths and room for development are identified, and backlog refinement is where adjustments to the product backlog are made for the upcoming sprint. These meetings take the form of business

interactions. The actors involved in the Sprint review meeting are stakeholders and Developers. The Scrum Master is also involved, having a Business role. Actors involved in the sprint retrospective are the Product Owner and the Developer. The actors involved in the backlog refinement are the Developers, the Product owners, and the Scrum Master. The meetings are connected with a flow relation as they transfer knowledge and feedback from the one before. After the meetings, the sprint backlog gets adjusted. It has an association relation between it and the meetings. Hereafter, a final sprint review (Business interaction) takes place where the Developers conduct the final touches, and once everything is in place, the product is ready to be delivered. The Product Owner is primarily involved in the release phase. After release, the Scrum team monitor the system and the product to ensure that everything is working as expected and free from mistakes. The monitoring is a technology event connected to the product by a flow relation.

DevOps, DevSecOps, Scrum

Modelling the methodologies and frameworks selected showed that they are well-organised mechanisms that can achieve product delivery efficiently. However, some similarities and differences have come to light.

Product backlog

DevOps, DevSecOps, and Scrum, start the project with a product backlog to discuss and draw the steps required to attain the project goals successfully. However, DevSecOps include a Security Analyst to assess and prioritise security risks and identify security requirements.

Meetings

Another aspect recognised is that meetings play a significant role in the work organisation in the Scrum framework. That is not the case for DevOps or DevSecOps. In Scrum, multiple meetings occur daily to ensure that requirements are met, problems are resolved, and goals are achieved.

Meetings occur before the day's code, after the day's development, after the incremental development, and after all backlog adjustments occur before reaching the release phase. In DevOps and DevSecOps, the development team, operations team, and security team (in DevSecOps only) collaborate throughout the product development lifecycle. However, no formal meeting unfolds. Also, unlike DevOps and DevSecOps, stakeholders are involved at a certain point in the Scrum framework, giving product feedback to ensure the project is on the right track. Overall, meetings are more common and relied on in Scrum than DevOps and DevSecOps, as well as stakeholder feedback.

Security

It was very clear from the modelling that DevSecOps incorporates security measures more than DevOps and Scrum. The lack of prioritisation of security in DevOps, increased cyber threats, and pressure from legal and governmental entities have prompted organisations to demand greater dedication to security. That is why a security layer was added to DevOps and has the name of DevSecOps. The product phases between the two are strongly similar. However, after the addition of the security team, extra security layers were included: Before the beginning of the project, during CI, after the coding, and before and after the product release. In Scrum, security measures occur daily after each sprint, allowing the code to stay secure and up-to-date.

Nonetheless, like DevOps, no security team is assigned to take care of the overall security of the project. Developers take care of security measures, making the security level less effective and reliable than DevSecOps. In the main, the code goes through multiple security layers in DevSecOps than in DevOps or Scrum.

Automation

Automation tools are in use in all frameworks and methodologies in question. Using automation tools is vital as it helps the teams and development companies automate their software whenever needed. That helps reduce the human intervention requirement, leading to greater speed, efficiency, and credibility (Testim, 2020). In all three, CI is used so Developers can work on the project simultaneously and merge their code into a shared repository accessible to the whole team. Additionally, DevOps and DevSecOps use automation tools in CI/CD during deployment and the monitoring phase.

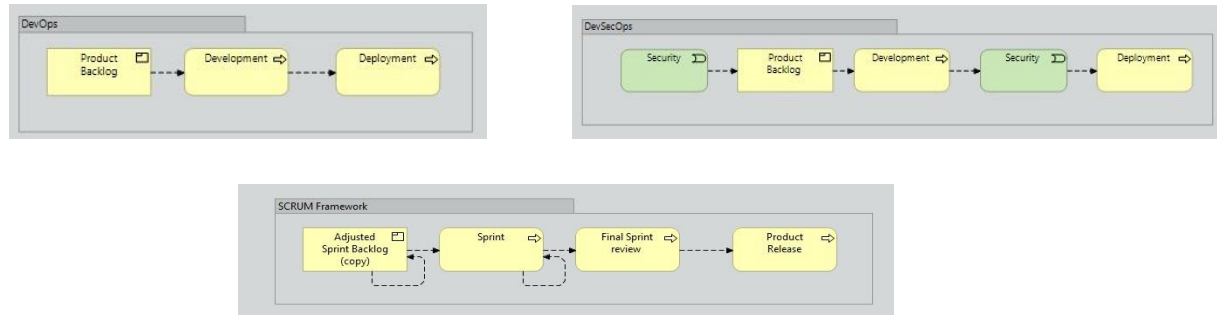
On top of that, automation is well relied on in DevSecOps to cover security tasks such as triggering code tests after code is pushed during CI and throughout system configuration. Like DevSecOps, security measures are automated after the code is implemented and written by the Developers in Scrum. DevSecOps uses more automation tools than Scrum and DevOps, making it less prone to human error.

Product lifecycle

Analysis is based on the models below and the models in Appendices 3, 4, and 5. DevOps, DevSecOps, and Scrum embrace continuous improvement, aiming to enhance processes and layers to reach the goals of the project being worked on. DevOps's Product Life cycle (PLC) is relatively fast as the methodology stresses shortening lead times and increasing deployment frequency. The PLC of DevSecOps is slightly longer as it incorporates security practices throughout the development phases. In comparison, Scrum is an agile framework that operates in sprints lasting more or less two weeks. It has a structured framework focusing on delivering shippable products at the end of each sprint. Its PLC cannot be compared to DevOps or

DevSecOps as it depends on specific needs, requirements, and market changes.

Figure 1: PLC of DevOps, DevSecOps, and Scrum using ArchiMate



Conclusion

This paper investigates the similarities and differences between software development methodologies and frameworks: DevOps, DevSecOps, and Scrum. A preliminary literature review gave a thorough description of each of these methods used by software development companies and showed their main focus while introducing their structure and phases implemented.

To visualise the structure and envision the product development lifecycle of the methods in question, an Enterprise Architecture modelling language called ArchiMate was used. Three models were created for each of them. Modelling these frameworks and methodologies showed that DevSecOps and DevOps share the same foundation. However, DevSecOps extends DevOps by including security practices and layers throughout the development lifecycle. It also showed that each method has different ways of organising the product development lifecycle with its system and practices. Scrum has different priorities than DevOps and DevSecOps as it is an agile framework that aims to adapt to the changing requirements of the product and market and continuously improve.

Directions for future research

The value of this study is that, for the first time, an attempt was made to model three different frameworks and methodologies using the EA approach. However, there are also limitations. First, DevOps, DevSecOps, and Scrum are dynamic and adaptive to changes in the market and requirements. Traditional EA models may not apprehend the evolving nature of these methods. Second, the methods in question are well known for the collaboration among multiple teams and everyday interaction on specific processes. EA models might not effectively capture communication patterns and team dynamics among them. To understand more how agile frameworks and methodologies are compared, the analysis can be carried out by comparing Kanban, Extreme programming, and Feature-Driven Development using EA (ArchiMate).

References

- AWS. 2022. "What is Scrum?" <https://aws.amazon.com/what-is/scrum/#:~:text=Scrum%20is%20a%20management%20framework,experience%2C%20and%20adapt%20to%20change>.
- Azham, Z. Ghani, I, and Ithnin N. 2011. "Security backlog in Scrum security practices." Malaysian Conference in Software Engineering. Johor Bahru, Malaysia: IEEE. pp. 414-417.
- Chen, T, and Suo, H. 2022. "Design and Practice of Security Architecture via DevSecOps Technology." 13th International Conference on Software Engineering and Service Science (ICSSESS), Beijing, China: IEEE. 2022. pp. 310-313.
- Cowell, C, Lotz, N, and Timberlake C. 2023. "Automating DevOps with GitLab CI/CD Pipelines: Build efficient CI/CD pipelines to verify, secure, and deploy your code using real-life examples." Birmingham: Packt Publishing.
- Faniran, V. T, Badru A, and Ajayi N. 2017. "Adopting Scrum as an Agile approach in distributed software development: A review of literature". 1st International Conference on Next Generation Computing Applications (NextComp). Mauritius:IEEE. pp. 36-40.
- Hoogervorst, Jan. 2003. "Enterprise Architecture: Enabling Integration, Agility And Change." International Journal of Cooperative Information Systems. 13(03):213-233
- IBM. 2022. "What is DevSecOps?" <https://www.ibm.com/topics/devsecops>.
- Macdonald, Neil, and Head Ian. 2016. "DevSecOps: How to Seamlessly Integrate Security Into DevOps." Gartner Journal.
- Mary Sánchez-Gordón and Ricardo Colomo-Palacios. 2020. "Security as Culture: A Systematic Literature Review of DevSecOps." In Proceedings of the ACM 42nd International Conference on Software Engineering Workshops (ICSEW'20). Association for Computing Machinery, New York, NY, USA: IEEE. 266–269.
- Matthies, C. 2019. "Feedback in Scrum: Data-Informed Retrospectives." ACM 41st International Conference on Software Engineering: Companion Proceedings (ICSE-Companion), Montreal, QC, Canada: IEEE. pp. 198-201.
- Mohan, V and Othmane, L. B. 2016. "SecDevOps: Is It a Marketing Buzzword? - Mapping Research on Security in DevOps." 11th International Conference on Availability, Reliability and Security (ARES). Salzburg, Austria: IEEE.
- Myrbakken, H., Colomo-Palacios, R. 2017. DevSecOps: A Multivocal Literature Review. In: Mas, A., Mesquida, A., O'Connor, R., Rout, T., Dorling, A. (eds) Software Process Improvement and Capability Determination. SPICE 2017. Communications in Computer and Information Science, vol 770.

Oghojafor, BEA, Olayemi, O, and Oluwatula. 2012. "Attribution Theory and Strategic Decisions on Organizational Success Factors." *Journal of Management and Strategy*, Vol3, No 1:8

Olsson H, Bosch J, and Alahryia H. 2013. "Towards R&D as Innovation Experiment Systems: A Framework for Moving Beyond Agile Software Development." *Journal of a Scientific and Technical Publishing Company*.

Pennington Jakob. 2018. "The Eight Phases of a DevOps Pipeline." Texas: Medium.

Petrović, N, Cankar, M, and Luzar, A. 2022. "Automated Approach to IaC Code Inspection Using Python-Based DevSecOps Tool," 2022 30th Telecommunications Forum (TELFOR), Belgrade, Serbia: IEEE. pp. 1-4.

Popli R, and Chauhan, N. 2013. "A sprint-point based estimation technique in Scrum." *International Conference on Information Systems and Computer Networks*. Mathura, India: IEEE. pp. 98-103.

Putra, A. M, and Kabetta H. 2022. "Implementation of DevSecOps by Integrating Static and Dynamic Security Testing in CI/CD Pipelines." *International Conference of Computer Science and Information Technology (ICOSNIKOM)*, Laguboti, North Sumatra, Indonesia: IEEE. pp. 1-6.

Schwaber, K. 1997. "SCRUM Development Process." London: Springer.

Scrum. 2012. "3 Steps to an Effective Retrospective." <https://www.scruminc.com/wp-content/uploads/2014/05/Three-Steps-to-an-Effective-Retrospective.pdf>.

Shahin, Mojtaba, and Babar Ali. 2020. "ICSSP: On the Role of Software Architecture in DevOps Transformation: An Industrial Case Study." Republic of Korea: IEEE.

Shah, Sandip, and O'Grady Darragh. 2020. "Adaptable Enterprise Architecture for Times of Rapid Change." *Journal of Global Business*.

Sravan, Surabattina. 2023. "Significant Challenges to espouse DevOps Culture in Software Organisations By AWS: A methodical Review." Coimbatore: IEEE.

Suteeca, K and Ramingwong, S. 2016. "A framework to apply ISO/IEC29110 on SCRUM." *International Computer Science and Engineering Conference (ICSEC)* Chiang Mai, Thailand: IEEE. pp. 1-5.

Sutherland, Jeff. 2006. "The First Scrum: How Scrum provides energy, focus, clarity, and transparency to project teams developing complex systems." Interview with Dr. Jeff Sutherland, CTO of PatientKeeper, Inc. California: EWork-Out.

The Open Group. 2016. An Introduction to the ArchiMate 3.0 Specification. <https://api-ir.unilag.edu.ng/server/api/core/bitstreams/985f6a6c-7ae4-4092-acc1-b82d6979abe1/content>

The Open Group. 2018. "The TOGAF Standard". <https://www.opengroup.org/togaf>

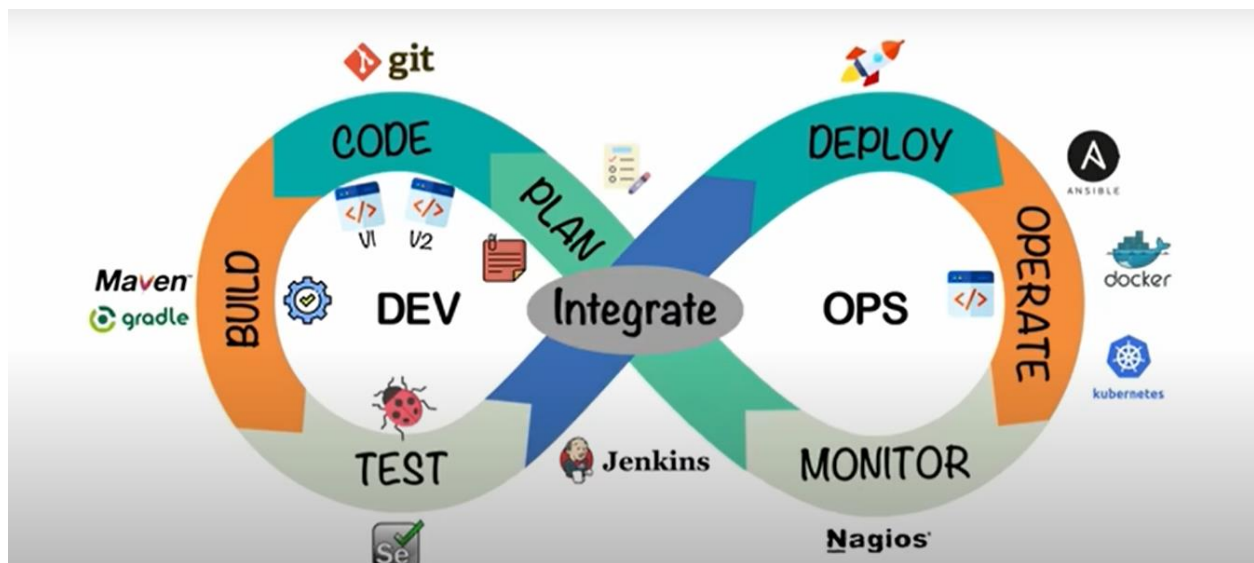
Tyagi, S, Akshit Raj. 2022. "ICI: Lightweight Review: Challenges and Benefits of Adopting DevOps." Noida: IEEE.

Vega, Florencia & Rodríguez, Guillermo & Rocha, Fabio & dos Santos, Rodrigo. (2022). Scrum Watch: a tool for monitoring the performance of Scrum-based work teams. JUCS - Journal of Universal Computer Science. 28. 98-117. 10.3897/jucs.67593.

Virmani Manish. 2015. "INTECH: Understanding DevOps & bridging the gap from continuous integration to continuous delivery." Sao Paulo: IEEE.

Appendix




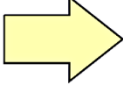




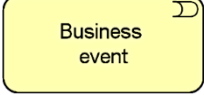



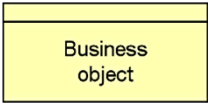
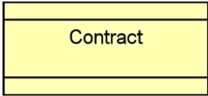

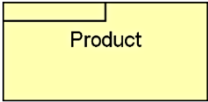
Appendix 1 - DevOps model



Appendix 2 – ArchiMate Representations

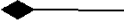
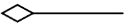

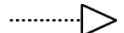



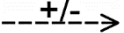


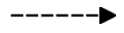

A- Business Layer Elements

| Element | Description | Notation |
|------------------------|--|-----------------------------------|
| Business actor | Represents a business entity that is capable of performing behavior. | <div>Business actor</div> |
| Business role | Represents the responsibility for performing specific behavior, to which an actor can be assigned, or the part an actor plays in a particular action or event. | <div>Business role</div> |
| Business collaboration | Represents an aggregate of two or more business internal active structure elements that work together to perform collective behavior. | <div>Business collaboration</div> |

| Element | Description | Notation |
|----------------------|---|--|
| Business interface | Represents a point of access where a business service is made available to the environment. |   |
| Business process | Represents a sequence of business behaviors that achieves a specific result such as a defined set of products or business services. |   |
| Business function | Represents a collection of business behavior based on a chosen set of criteria (typically required business resources and/or competencies), closely aligned to an organisation, but not necessarily explicitly governed by the organisation. |   |
| Business interaction | Represents a unit of collective business behavior performed by (a collaboration of) two or more business actors, business roles, or business collaborations. |   |
| Business event | Represents an organisational state change. |   |
| Business service | Represents explicitly defined behavior that a business role, business actor, or business collaboration exposes to its environment. |   |
| Business object | Represents a concept used within a particular business domain. |  |
| Contract | Represents a formal or informal specification of an agreement between a provider and a consumer that specifies the rights and obligations associated with a product and establishes functional and non-functional parameters for interaction. |  |
| Representation | Represents a perceptible form of the information carried by a business object. |  |
| Product | Represents a coherent collection of services and/or passive structure elements, accompanied by a contract/set of agreements, which is |  |


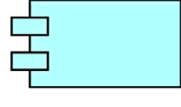





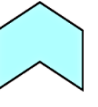

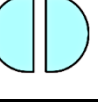
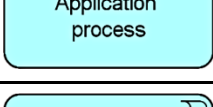
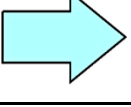
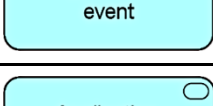

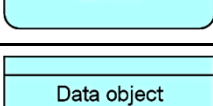
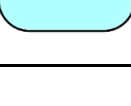

| Element | Description | Notation |
|---------|---|----------|
| | offered as a whole to (internal or external) customers. | |

B- Relationships

| Structural Relationships | | Notation | Role Names |
|--------------------------|---|--|---|
| Composition | Represents that an element consists of one or more other concepts. |  | ← composed of → composed in |
| Aggregation | Represents that an element combines one or more other concepts. |  | ← aggregates → aggregated in |
| Assignment | Represents the allocation of responsibility, performance of behavior, storage, or execution. |  | ← assigned to → has assigned |
| Realisation | Represents that an entity plays a critical role in the creation, achievement, sustenance, or operation of a more abstract entity. |  | ← realises → realised by |
| Dependency Relationships | | Notation | Role Names |
| Serving | Represents that an element provides its functionality to another element. |  | ← serves → served by |
| Access | Represents the ability of behavior and active structure elements to observe or act upon passive structure elements. |   | ← accesses → accessed by |
| Influence | Represents that an element affects the implementation or achievement of some motivation element. |  | ← influences → influenced by |
| Association | Represents an unspecified relationship, or one that is not represented by another ArchiMate relationship. |  | associated with ← associated to → associated from |
| Dynamic Relationships | | Notation | Role Names |
| Triggering | Represents a temporal or causal relationship between elements. |  | ← triggers → triggered by |
| Flow | Represents transfer from one element to another. |  | ← flows to → flows from |
| Other Relationships | | Notation | Role Names |
| Specialisation | Represents that an element is a particular kind of another element. |  | ← specialises → specialised by |

| Relationship Connectors | | Notation | Role Names |
|-------------------------|---|---------------------|------------|
| Junction | Used to connect relationships of the same type. | ● (And) Junction | |
| | | ○ Or Junction | |

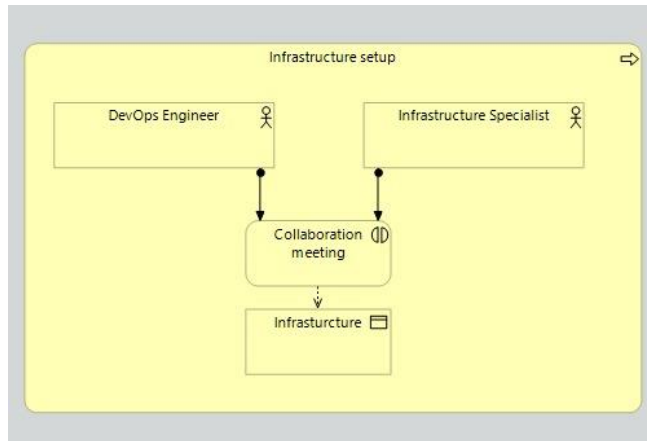
C- Application Layer Elements

| Element | Definition | Notation |
|---------------------------|--|--|
| Application component | Represents an encapsulation of application functionality aligned to implementation structure, which is modular and replaceable. |   |
| Application collaboration | Represents an aggregate of two or more application internal active structure elements that work together to perform collective application behavior. |   |
| Application interface | Represents a point of access where application services are made available to a user, another application component, or a node. |   |
| Application function | Represents automated behavior that can be performed by an application component. |   |
| Application interaction | Represents a unit of collective application behavior performed by (a collaboration of) two or more application components. |   |
| Application process | Represents a sequence of application behaviors that achieves a specific result. |   |
| Application event | Represents an application state change. |   |
| Application service | Represents an explicitly defined exposed application behavior. |   |
| Data object | Represents data structured for automated processing. |  |

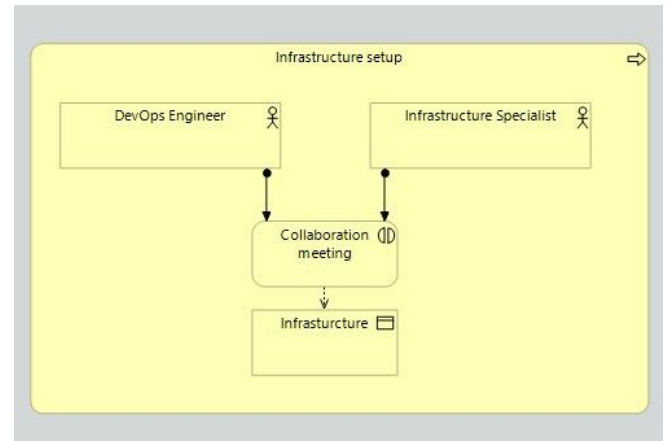
Source: <https://pubs.opengroup.org/architecture/archimate31-doc/chap08.html>
<https://pubs.opengroup.org/architecture/archimate31-doc/chap09.html>
<https://pubs.opengroup.org/architecture/archimate31-doc/chap05.html>

Appendix 3 - DevOps Modeling

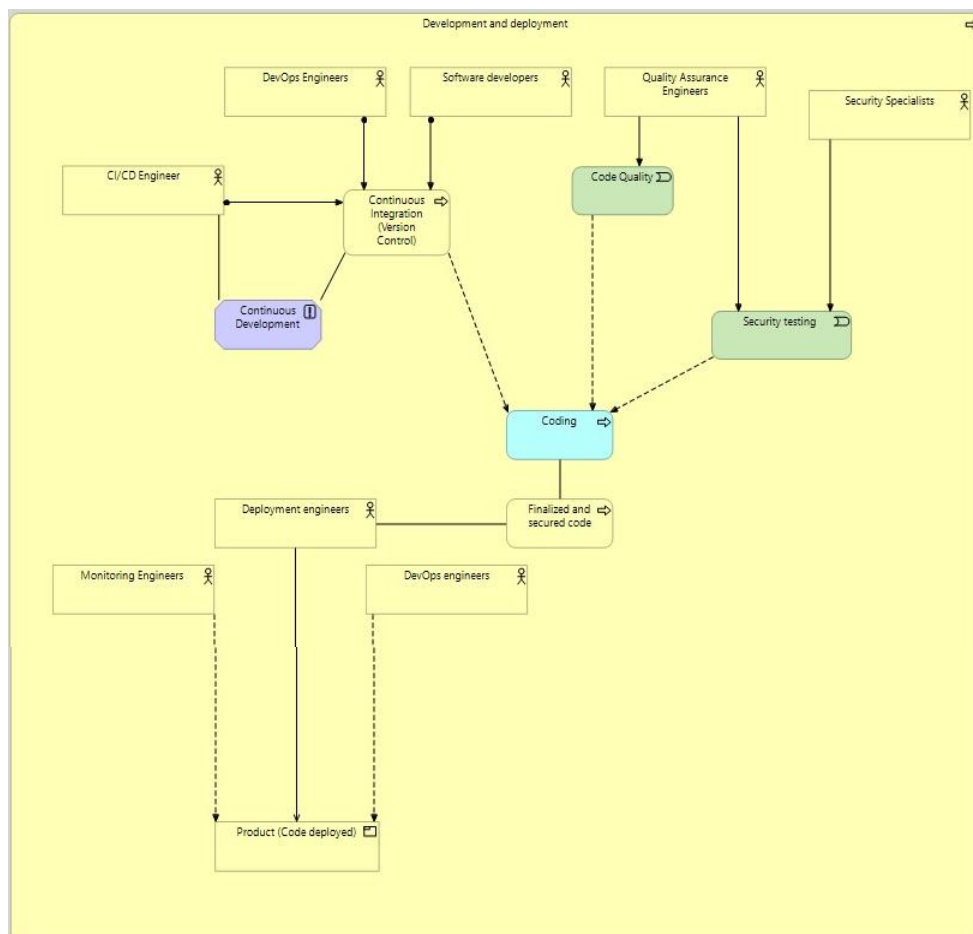
1



2

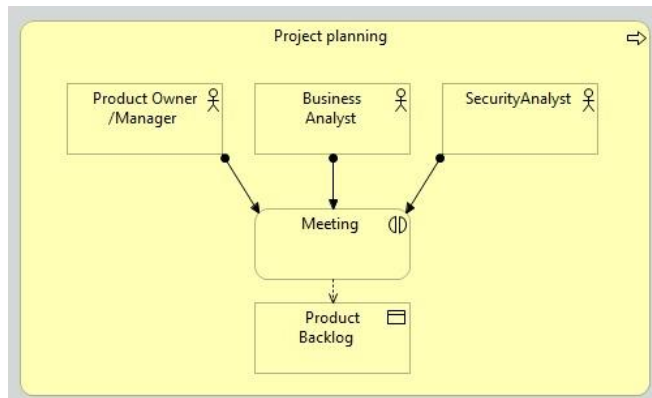


3

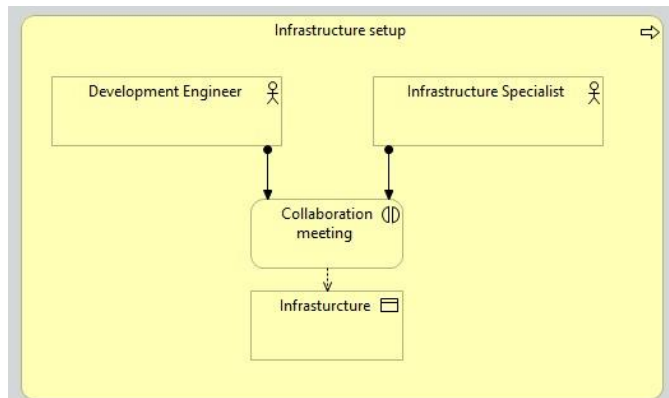


Appendix 4 - DevSecOps Modeling

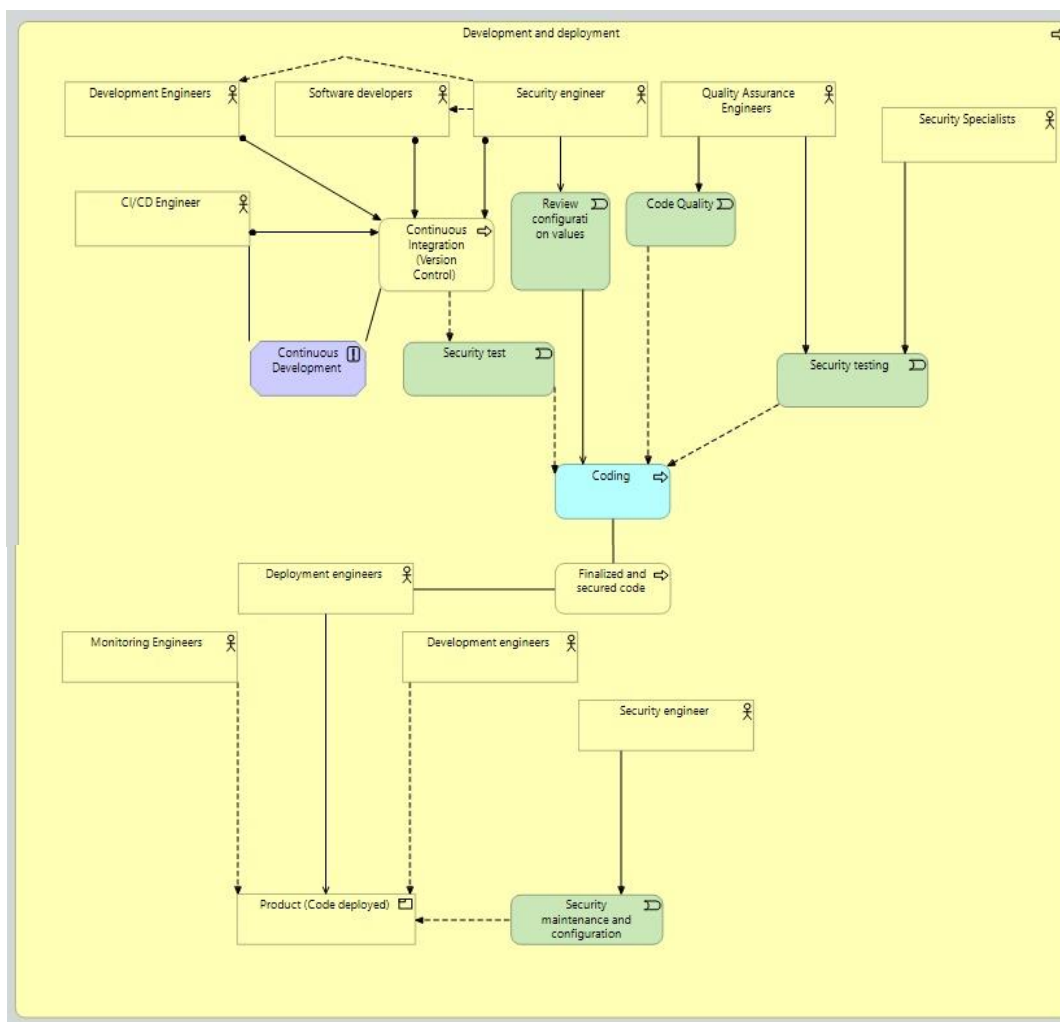
1



2



3



1

