Lesson Learnt by Using DevOps and Scrum for Development a Traceability Software

Dimitrios Salmas, Giannis Botilias, Jeries Besharat, and Chrysostomos Stylios(✉)

Department of Informatics and Telecommunications, University of Ioannina, Kostakioi, 47150 Arta, Greece
{salmasdimitris,jbotilias,jeries.besharat}@kic.uoi.gr, stylios@uoi.gr

Abstract. Agile has significantly impacted the software development lifecycle by introducing methodologies such as Scrum, Extreme Programming, and Lean Software Development. Recently, DevOps approach has attracted and gained a wide interest of the software development society. DevOps provides a set of principles that enables Continuous Development and Continuous Integration of a system. This paper presents a case study where it is designed and developed a food traceability software for a Greek meat company; it discusses the lesson learnt by applying the DevOps principles for the software development and by using the Scrum methodology for management purposes.

Keywords: DevOps · Agile · Scrum · Software development

1 Introduction

DevOps is a set of principles and practices that are used to bridge the gap between the Development and IT operation teams [1, 2]. DevOps' primary objective is to unify both teams to reduce committing a change to a system while ensuring high quality. DevOps' principles are focused on improving the teamwork, communication, and collaboration of developers and operators [3]. Based on those principles, the DevOps focuses on the following four main concerns [4]:

- Quickly getting a change into production.
- Finding errors through automated testing.
- Reducing or eliminating errors that occur during deployment.
- Quickly finding and repairing faults in the system.

The DevOps approach and culture can support Agile's delivery cycle from inferring and providing the design specifications and documents to comply with the Continuous Development and Continuous Integration of the DevOps culture.

Agile software development principles, values and practices are required for successful DevOps adoption [5] as Agile methodologies come to give a more effective way of managing projects and developing products and services.

Agile Methodologies are a group of software development methods that are based on iterative and incremental development [6]. They have been developed to address the problems faced by the conventional model and to offer project teams multiple possibilities during the development process [7]. The methodologies for software development diversify and support a gradual development of requirements. This manner of incremental requirements refinement further refines the design, coding and testing at all stages of production activity. This project management style allows for adaptive planning, iterative & evolutionary development, rapid and flexible response to change and promote communication, which are the four major characteristics that are fundamental to all agile methodologies [6].

In other words, Agile methodologies are used to achieve higher quality software in a shorter period of time, self organizing teams, customer collaboration, less documentation and reduced time to market [6]. To achieve these goals, there are a lot of agile methodologies that are used. Most popular are the Scrum, Extreme Programming (XP), Kanban, Crystal Methods and Feature Driven Development (FDD) [8]. This study will be focused on Scrum Methodology, which is specifically designed to handle rapidly changing business requirements.

This paper presents the main characteristics of DevOps approach in collaboration with the best framework of Agile methodology, the Scrum. It aims to describe how the development aspects that Scrum offers and the rapid delivery aspects that DevOps offers could be combined in the case of developing a food traceability software [9]. It presents the evaluation results of applying the abovementioned methodologies and their impact to software development team and the software itself.

2 Case Study

This paper is focusing on a cattle/beef traceability software application developed for a Greek meat market to track the whole meat chain from farm to the final product sold on the shelf of the supermarket. The application consists of a desktop and mobile application and is based on a cloud platform and web technologies.

This web application was developed with a DevOps approach and the agile methodology as a management tool. Different users in different places in the traceability chain will submit data over the mobile or desktop application. The software is developed in such a way that provide access to all the details crucial to the final product's backward and forward traceability.

3 Methodology

The applied and described methodology adapts Agile approach in the DevOps culture. In this methodology, Scrum has been chosen as the main Agile framework that is used to cover the managerial aspect of the software development lifecycle while DevOps is used to support the rapid deployment of the functionalities.

Given below are the stages of the software development lifecycle that were used in the use case:

- Management
- System integration testing
- Testing user acceptance test
- Deploy monitoring.

3.1 Management

Scrum

The Scrum methodology focuses greater on management of the development process than software program coding strategies. Scrum moves a project forward by improving communication between team members and breaking the work into short time frames ("sprints") that run usually from one to four weeks. Although Scrum works effectively on both small and large projects [8], Scrum is mainly a software development process for small teams. Researchers have shown that small teams that work independently are more effective [10]. In Scrum methodology, teams work as tight, integrated units with each team member playing a well-defined role and the whole team focusing on a single goal.

Sprint

A sprint produces a visible, usable, deliverable product that implements one or more user interactions with the system. The key idea behind each sprint is to deliver valuable functionality. Sprint is one-time boxed iteration of the continuous software improvement and development cycle and it has consistent duration throughout a development effort. The consistent duration means that the end date for a sprint does not change [10].

In the case study, the sprints were weekly and during the sprint the team has hold frequent Scrum meetings via Slack tool. These meetings were attended by all members of the team, including those who worked remotely. The purpose of these meetings was to bring all team members closely to talk about problems that arose during implementation. The duration of Scrum meetings was short (15–20 min) and kept everyone informed of team progress and obstacles.

At the end of a sprint, a final meeting is scheduled in which all new information from the just completed sprint is reported. At this meeting, anything can be changed. Work can be added, eliminated, or reprioritized. In continuation, all project team participate to a meeting with all stakeholders, including high-level management, customers, and customer representatives to discuss the information of the sprint and suggest possible changes. These meetings usually are triggering the initiation of a new cycle of sprints.

Single Source Repository

The DevOps team used a Single Source Repository for code management and version control in order to achieve Continuous Integration and Continuous Delivery. All team members had access to the repository and they were permitted to keep track of development progress. Each new feature, that needs to be developed, had its own branch in the repository and it passed through three phases. Each phase had a separate identical environment and a branch in the repository:

- The development branch: In this branch all the new features were initially implemented and passed a unit testing before being pushed to the stage branch.
- Stage branch: This branch was identical to the production branch, and all features implemented passed user acceptance tests before they were pushed to the production branch.
- Production branch: This branch contains the code of the working version of the application which is constantly online and monitored. Any features committed in this branch were already tested and ready to use.

In order to work in a scrum environment and produce code, without interrupting the final product, the team followed some rules:

- There was no time limit on committing new changes in development mode.
- No changes in the codes of stage and production branch.
- The development branch never pushes changes directly to the production branch.
- Each environment had its own database to ensure the integrity of the data.

3.2 System Integration Testing

System Integration Testing (SIT) is the overall testing procedure of the whole system, which is composed of many sub-systems. The main goal of SIT is to ensure that all software module dependencies are functioning properly and the data integrity is preserved between distinct modules of the whole system. Integration testing is an important part of the testing process and is one of the most common methods for assuring the quality of complex computer software systems [11].

The development of the system was implemented in an identical environment for the production and upholding the Continuous Integration philosophy of DevOps. The development of new features was integrated according to Agile's test-driven development. At the start of each feature, a unit testing was developed. The developers updated the codes until the new feature passed the unit testing. After the code passed the unit testing, its quality and technical debt were automatically inspected. Afterward the developers refactored the existing code until its quality was at a satisfactory level and the technical debt was as low as possible.

Before integrating the feature to the stage phase for the user acceptance test, it passed through the system integration testing.

3.3 Testing User Acceptance Test

In the stage environment, the developer teams were conducting user acceptance testing of the software's new features. User Acceptance Testing (UAT) involves validating software in a real setting by the intended audience. The aim is not so much to check the defined requirements but to ensure that the software satisfies the customer's needs. Agile methodologies put stringent demands on UAT, if only for the frequency at which it needs to be conducted due to the iterative development of small product releases [12].

At the end of each sprint, any features that meet the requirements were deployed to the production environment.

3.4 Deployment and Monitoring

In this case study, the team was working in a test environment similar to the production environment. As part of the Continuous Deployment, the team did create a Webhook service that was triggered on the successful completion of the Continuous Integration and Testing stage.

The production environment is constantly monitored and provides to the team useful information regarding maintenance and measurements of the application operation. The continuous monitoring of the software produces live data as well as logs that are used for further analysis. The analysis includes event tracking, profiling, and performance issues.

4 Tools

4.1 Scrum Management

Trello is a collaboration tool that is used for organizing projects and is created by Atlassian. The Trello is being used by the team for project management. Each sprint has its own list and each features its own card with descriptions on time estimation and other useful information attached to it. Each member of the team is attached to the cards to allow the project manager to have a better overview of the team members' workload [13].

4.2 ChatOps

Due to the covid19 pandemic, the team is working remotely. Therefore, a need for better and faster communication between its members emerged. The DevOps team adopted Slack as the default communication platform. Slack is a channel-based messaging platform that provides a connection to different software such as google calendar and Trello [14].

4.3 Source Management

For Source management, the team used Git and more specifically GitHub as its repository. GitHub is a user interface with various features that are based on the Git command-line tool. GitHub contains everything that is needed from the developers' team such as version control, issue tracking, documentation, and status dashboards [15].

4.4 Source Code Quality Management

SonarQube uses for checking code coverage and provides information regarding the source code quality. SonarQube is an open-source software quality assurance tool released by SonarSource. It supports over 20 languages and it has various source code metrics, coding rules, violations, code duplication and provides an estimation of the technical debt [16].

4.5 Deployment Tool

The Pm2 is used to manage and monitor the Node.js server. Pm2 is process management for Node.js applications that also provides real-time data or monitoring [17]. The Pm2 monitoring integration provides vital data concerning the health and status of the application as well as real-time logs, all in one dashboard.

4.6 Unit Testing

Mocha and Chai have been selected for the unit testing. Mocha is a JavaScript test framework running on NodeJS and it can be used with Chai, a BDD/TDD assertion library [18, 19]. They are used for the test-driving development of the software. Loader.io is used to test the system response time. Loader.io is a load testing service that allows the developers to stress test API and web applications [20].

5 Learning Outcomes

This section is presenting the main learning outcomes and how there were improved the capabilities of the software development team.

5.1 DevOPS

During the development procedure, the software development team learned how to use the methodology and tools for Continuous Development and Continuous Integration. The capabilities related to Continuous Development and Continuous Integration is mainly provided by DevOps approach paired with Agile's Sprint procedure that provide the Project Manager's ability to quickly monitor the team. By following the four stages of the software development life cycle introduced and adopted in the development of this application, the team managed to successfully adapt to the DevOps culture. The team members already used GIT, and they adapted with ease on using GitHub as the main Source Management where they created a new branch for each feature without disrupting the flow of software development. The usage of a source management tool such as Git supported the aim of a successful Continuous Software Development and the integration of new features, based on the requirements and tasks assigned at the beginning of each Sprint.

At the initial phase of the project was difficult for the software development team to follow the adopted methodology and the proposed tools that were used for Automatic deployment, monitoring, quality management and unit testing, but after three weeks, the team had fully used the approach. At the final stages of the application development, the team was asked to evaluate the new procedure and they concluded that it is faster and easier to use those to test their code and deploy the new features.

5.2 Agile

This specific project was the first time for the development team that they were introduced to Agile's world. Thus, it was evaluated the followed procedure and it was concluded that its adapting Agile methodology for software development produced positive outcomes both for the whole team and for each individual programmer. More specifically, the Scrum methodology trained the team and it increased teamwork, the level and quality of collaboration and communication; especially the Scrum meetings contributed to a perfect synergy between project management and software development skills. In addition to this, the Scrum methodology provided opportunities to each programmer to upgrowth programming skills in many aspects such as in server and client-side programming, development, programming languages, tooling and task management as well as software problem solving. Furthermore, one of the most important positive outcomes of the adopted Scrum methodology is the individual responsibility shown by each member of the team at critical points of the project, where they accepted their tasks without transferring his responsibilities to the other members. In conclusion, adopting Scrum methodology led at producing higher quality software and deliverables and in achieving the final goal consistently and accurately.

6 Conclusion

Due to the competitiveness of the software application field, programmers are under constant pressure to deliver software updates and include advanced features in the minimum possible time. In the specific case study, the development team was introduced for first time in adopting a balanced combination between DevOps culture and Scrum methodology that it lead to achieving better results such as enabling the rapid inclusion of new features and ensuring high quality. Here it was presented the main methodology and tools that the software team used and adopted for the specific case for developing food traceability software.

Acknowledgment. This research work is funded by the Operational Programme "Epirus" 2014–2020, under the project "Advanced traceability of Epirus meat to improve productivity, quality and hygiene by using Business Intelligence Systems – EUKREAS", Co-financed by the European Regional Development Fund (ERDF).

References

1. Govil, N., Saurakhia, M., Agnihotri, P., Shukla, S., Agarwal, S.: Analyzing the behaviour of applying agile methodologies & DevOps culture in e-Commerce web application. In: 2020 4th International Conference on Trends in Electronics and Informatics (ICOEI), vol. 48184 (2020). https://doi.org/10.1109/icoei48184.2020.9142895
2. Jabbari, R., Ali, N., Petersen, K., Tanveer, B.: What is DevOps?: A systematic mapping study on definitions and practices. J. Softw. Evol. Process **1–11** (2016). https://doi.org/10.1145/2962695.2962707

3. Jabbari, R., bin Ali, N., Petersen, K., Tanveer, B.: Towards a benefits dependency network for DevOps based on a systematic literature review. J. Softw. Evol. Process **30** (2018). https://doi.org/10.1002/smr.1957
4. Bass, L.: The software architect and DevOps. IEEE Softw. **35**, 8 (2018). https://doi.org/10.1109/ms.2017.4541051
5. Lwakatare, L., Kuvaja, P., Oivo, M.: Relationship of DevOps to agile, lean and continuous deployment. Product-focused software process improvement, pp. 399–415 (2016)
6. Kumar, G., Kumar Bhatia, P.: Impact of agile methodology on software development process. Int. J. Comput. Technol. Electron. Eng. (IJCTEE) **2**, 2249–6343 (2012)
7. Mohammad, S.: DevOps automation and Agile methodology. SSRN Electron. J. **5**, 946–949 (2017). https://doi.org/10.1729/Journal.24060
8. Livermore, J.: Factors that significantly impact the implementation of an agile software development methodology. J. Softw. (2008). https://doi.org/10.4304/jsw.3.4.31-36
9. Samarawickrama, S., Perera, I.: Continuous scrum: a framework to enhance scrum with DevOps. In: 2017 Seventeenth International Conference on Advances in ICT for Emerging Regions (ICTer) (2017)
10. Rising, L., Janoff, N.: The Scrum software development process for small teams. IEEE Softw. **17**, 26–32 (2000). https://doi.org/10.1109/52.854065
11. Jin, Z., Offutt, A.J.: Coupling-based criteria for integration testing. Softw. Test Verif. Reliab. **8**(3), 133–154 (1998)
12. Otaduy, I., Diaz, O.: User acceptance testing for Agile-developed web-based applications: empowering customers through wikis and mind maps. J. Syst. Softw. **133**, 212–229 (2017). https://doi.org/10.1016/j.jss.2017.01.002
13. What is Trello? - Trello Help. In: Help.trello.com (2020). https://help.trello.com/article/708-what-is-trello.
14. What is Slack? In: Slack Help Center (2020). https://slack.com/intl/en-gr/help/articles/115004071768-What-is-Slack-.
15. Bleiel, N.: Collaborating in GitHub. In: 2016 IEEE International Professional Communication Conference (IPCC) (2016). https://doi.org/10.1109/ipcc.2016.7740497
16. Barta, B., Manz, G., Siket, I., Ferenc, R.: Challenges of SonarQube plug-in maintenance. In: 2019 IEEE 26th International Conference on Software Analysis, Evolution and Reengineering (SANER) (2019). https://doi.org/10.1109/saner.2019.8667988
17. PM2. https://pm2.keymetrics.io/
18. The fun, simple, flexible JavaScript test framework. In: Mocha. https://mochajs.org/
19. Chai Assertion Library. In: Chai. https://www.chaijs.com/
20. API Docs. In: Loader.io Documentation. https://docs.loader.io/