DEVOPS FOR MANUFACTURING SYSTEMS: SPEEDING UP SOFTWARE DEVELOPMENT

Blüher, Till (1);
Maelzer, Daniel (2);
Harrendorf, Jessica (2);
Stark, Rainer (1)

1: Technische Universität Berlin, Industrielle Informationstechnik;
2: Mercedes-Benz AG, Maintenance Engineering

ABSTRACT

The increasing importance of software as an essential functional provider in products and processes requires that companies master the development capabilities to continiously and quickly deliver high-quality software features. DevOps (acronym for Development and Operations) is an essential approach to these capabilities, which has so far been used predominantly in software-driven companies. This paper investigates the conditions under which DevOps can also be used in the industrial context of series manufacturing in order to be able to steadily develop and provide software features used in these environments (e.g. for monitoring and maintenance of manufacturing machines). A concept for DevOps for manufacturing systems was developed based on current best practices and the specific situation in the industrial company. The concept was then implemented and validated with experts on the basis of initial development cycles, demonstrating the usefulness of DevOps for manufacturing systems.

Keywords: DevOps, Industry 4.0, Case study, Design practice

Contact:
Blüher, Till
Technische Universität Berlin
Germany
till.blueher@tu-berlin.de

Cite this article: Blüher, T., Maelzer, D., Harrendorf, J., Stark, R. (2023) 'DevOps for Manufacturing Systems: Speeding Up Software Development', in *Proceedings of the International Conference on Engineering Design (ICED23),* Bordeaux, France, 24-28 July 2023. DOI:10.1017/pds.2023.148

1 INTRODUCTION AND MOTIVATION

Software is becoming increasingly important in application domains such as mechanical engineering or production, as it is an essential functional carrier in the context of Industrie 4.0 (Stark, 2021). In pure software-based industries such as online commerce, established methods and tools for the continuous development, integration and deployment (CI/CD) of software have long been used (Ståhl and Bosch, 2013). The IT technical perspective on CI/CD techniques in conjunction with agile process models has given rise to the DevOps approach (Bass *et al.*, 2015). CI covers the testing of software, CD the deployment on a server, and DevOps additionally includes the steps of operation and monitoring. DevOps is a synthetic word consisting of Development and Operations and is defined differently in terms of its concrete scope (Lwakatare *et al.*, 2015). Dyck *et al.* (2015) synthesise a definition for DevOps as: "*DevOps is an organizational approach that stresses empathy and cross-functional collaboration within and between teams – especially development and IT operations – in software development organizations, in order to operate resilient systems and accelerate delivery of changes*."

However, the introduction of DevOps also repeatedly poses challenges for non-software based companies, because processes and ways of working have to be adapted, engineers and developers have to be trained for a new working mentality, new IT tools have to be introduced and integrated, and ultimately, this new infrastructure has to be operated (Leite *et al.*, 2020). Another dimension of challenges in the use of DevOps in the industrial context is the integration of these approaches into the existing IT, process and data landscape of industrial companies and their complex organizational structures (Gokarna and Singh, 2021). Mostly different monolithic systems are used there, and their modification and integration with new IT tools, such as those required for DevOps, present a major challenge from the point of view of information technology (interfaces, data formats, etc.) and organization (approvals, security policies, etc.) (Hasselbring *et al.*, 2019).

In this paper, a concept for DevOps for an industrial company to develop software used in the manufacturing environment is proposed and validated through implementation. The implementation enables the company to develop and deliver more complex software products faster and addresses the organisational and IT-technical constraints of industrial companies. In addition, continuous integration and development leads to a reduction in manual activities, lowering costs and error rates. Finally, the study provides insights into the challenges and opportunities of implementing DevOps solutions in the manufacturing sector.

2 STATE OF THE ART

2.1 DevOps tools, culture and cloud-infrastructure

DevOps manifests the principles and culture of agile software development, which feature a shortening of iterative software development cycles by allowing software elements to be developed, tested and rolled out incrementally (Fowler, 2001). Such agile development has been manifested in numerous process models (Scrum, Extreme Programming, Rational Unified Process, Dynamic Systems Development, etc.) and is IT-supported by DevOps (Abrahamsson *et al.*, 2017; Gustavsson, 2016). To implement DevOps, different tools are needed across the DevOps cycle from planning, coding, building, testing, releasing, deploying, operating and monitoring (Ebert *et al.*, 2016; Kersten, 2018; Macarthy and Bass, 2020). The tools fulfil a wide variety of functions and can be integrated via standardised interfaces and data formats (e.g. Rest, JSON). For DevOps to be used on an industrial level, the scalability of the tools and infrastructure (e.g., in terms of storage, computing power, global distribution of servers, latency) is also important. For this purpose, services from established cloud providers are often used, which provide hardware and software infrastructure as-a-service and charge on a usage basis. These services include, for example, the automated scaling of server clusters, monitoring of performance or the orchestration of deployed software services (Airaj, 2017).

2.2 Microservices and IoT-infrastructure within the production environment

Microservices are an IT architecture that provide software functionality as multiple separate services, in contrast to monolithic architectures, that can only run as a single code. In doing so, each microservice can use its individual stack of technology and forms around business functionalities (Hasselbring and Steinacker, 2017). Individual teams, with appropriate business domain knowledge

https://doi.org/10.1017/pds.2023.148 Published online by Cambridge University Press

are responsible for the overall development and deployment of the service and can use the optimal programming language, database, etc. to implement the business functionality as a microservice (Hasselbring and Steinacker, 2017). Communication between microservices takes place via lightweight interfaces (e.g., REST APIs) so that data managed separately by the microservices can be exchanged (Zimmermann, 2017). Advantages of the Microservice-Architecture are: flexibility, modularity, ability for evolution (Hasselbring and Steinacker, 2017), (Dragoni *et al.*, 2017). Microservices are provided as images in containers and can therefore also be referred to as containerized software (Rufino *et al.*, 2017).
While DevOps is state of the art in most software-based businesses (e.g. e-commerce), applications embedded in the physical world such as manufacturing plants, vehicles and other machines, DevOps is also increasingly being pursued as a software development approach (Wijaya *et al.*, 2019).
However, the development and operation of software in the context of embedded systems is significantly different from purely desktop-based applications. Among other things, errors in the software can lead to impairments of the physical system, which in the worst case can lead to accidents. The time component must be considered more strongly in the case of physical devices, and specialized domain knowledge is necessary when developing suitable software (Lwakatare *et al.*, 2016). In addition, an extended technology stack down to the physical machines is necessary to obtain the necessary connectivity. Various standards have already been developed for connecting machines (MQTT, OPC-UA etc.). A corresponding extended software stack for IoT is also necessary for software applications that are not deployed directly on machines, but which use machine data, since physical plants and products must be networked in order to be able to send relevant data (Roy *et al.*, 2016). This is particularly true in the production context, where legacy systems are often in use without the respective abilities for interconnection (Fisch *et al.*, 2018).

2.3 DevOps in the manufacturing context

Hasselbring *et al.* (2019) highlight the lack of knowledge and experience of legacy companies in using DevOps as a major impediment. They develop a framework for the implementation of DevOps in industrial companies focusing on a role model but less on the technical implementation of DevOps. Lwakatare *et al.* (2019) analyse five industrial companies regarding their approach to DevOps implementation. They found key success factors that promote successful DevOps implementation but do not describe a concrete implementation scheme in detail. Macarthy and Bass (2020) also researche DevOps implementation efforts in industrial companies and proposes a conceptual map linking DevOps tools to common challenges. They acknowledge the need for adapting DevOps concepts for applications involving physical products, but focus on the software industry otherwise and do not provide further guidance. Riungu-Kalliosaari *et al.* (2016) also conducted a case study with industrial companies using DevOps regarding benefits and challenges. They conclude that DevOps might not be suited for every industry, especially when there is a lack of technical knowledge or mindset in the workforce. They highlight the complexity of development and production environments in manufacturing as a major inhibitor of successful DevOps usage. Laukkarinen *et al.* (2017) examine the applicability of DevOps for medical devices, which are highly safety-relevant embedded devices. They find that certain regulatory standards (e.g. for approvals) forbid the usage of pure DevOps posing the challenge of developing the DevOps approaches in these domains. Gustavo Schmitz Albino (2022) present detailed research on an implementation infrastructure for DevOps to deploy machine learning applications in the manufacturing context.

3 METHODS

This section briefly explains the steps that were taken to create and validate the solution concept for implementing DevOps principles in an industrial company.
All research took place as embedded research throughout, meaning that the research and implementation was conducted on-site as an active part of the team in the company. This therefore includes participation in weekly team meetings, as well as free access to key knowledge repositories, stakeholders, and factory- as well as IT-assets, which allowed for in-depth elaboration and investigation of requirements and solution elements over a sustained period of time.

https://doi.org/10.1017/pds.2023.148 Published online by Cambridge University Press

3.1 Determination of the as-is situation of the DevOps pipeline

First, an understanding of the goals and current situation of software development with regard to processes and the IT and data landscape in the production context had to be gained. For this purpose, it was relevant to determine how current working methods are executed and which existing IT systems need to be integrated. To this end, interviews were conducted with six engineers and managers from the areas of software development and production. Each interview lasted about 60 minutes. In addition, an analysis of processes and documents were performed by analysing knowledge repositories of the company that contain, for example, process or IT tool descriptions. The findings on the As-Is situation were then summarized as needs for the DevOps concept.

3.2 Collection of requirements and synthesis towards the DevOps concept

In parallel, general requirements for DevOps were collected from the Best Practice literature and together with suppliers of the industrial firm analysing the As-Is situation in order to identify gaps and derive the requirements for DevOps in an industrial context.
The closure of these gaps was outlined in the description of the DevOps concept. During the collection of requirements and their synthesis into the target concept, continuous evaluation with experts from the industrial company and suppliers took place during regular team meetings as well as short coordination sessions during embedded activities. In total, the research period ran for six months with at least two coordination meetings of about 3 hours per two weeks and shorter daily meetings.

3.3 Implementation and validation in the production environment

The validation of the concept was carried out implementing the DevOps concept and using an exemplary software development cycle in a production context. For this purpose, the essential elements of the DevOps concept were implemented in order to determine an improvement in the software development processes. The evaluation took place by measuring lead times in the development of software features as well as by qualitative evaluation with experts before and after the introduction of the solution. The software used for validation is designed to compare the similarity of a production cycle with a predefined, high-quality reference cycle of a drilling machine tool based on several hundred characteristics (e.g. turning speed, temperature, time intervals) in order to detect deviations in the production step in real time. In case of significant deviations, the maintenance team is then notified via the software used for validation and can investigate the causes in the production system. The software does not access the machine directly, but the data stored on a server and is deployed on a server. Data is exported via machine-specific interfaces and software, that are not in the scope of the validation software.

4 RESULTS

In this research project, DevOps was set out to be implemented for software projects in an industrial production environment. The software applications used there, do not run on production equipment itself and do not interact with it directly (e.g., control of equipment), so that the special requirements of embedded software did not have to be considered. Instead, the software is server-based and is provided as containerized software in the sense of the described microservices. These applications often are called smart services and support employees during their work in factory operations (e.g., through analytics services). In the following, in accordance with the method described in Chapter 2, the industry-specific requirements for a DevOps pipeline are first synthesized. Then the concept for the target pipeline is presented and finally the validation results, based on the implementation of the concept, are described.

4.1 Requirements for the implementation of DevOps in industry

The requirements for DevOps in the industrial context were gathered from two perspectives. On the one hand, the current best practices in the literature (technology push) were reviewed. On the other hand, the specific requirements for continuous software development and operation in the company were determined in terms of the AS-IS situation in order to derive requirements from this (technology pull). Finally, a list of requirements was created in the synthesis and individual requirements were prioritized for the target concept to be implemented.

https://doi.org/10.1017/pds.2023.148 Published online by Cambridge University Press

4.1.1 Description of the AS-IS situation

The issues found resulting from the document analysis, interviews and group discussions are presented in the following as an excerpt with exemplary issues. They are structured by the traditional lifecycle phases of a software artefact (specification, implementation, integration and deployment, operation) (Table 1).

Table 1: Excerpt of AS-IS situation of the development and operations process of software for the production team over life cycle stages

Life Cycle Stage	Identified Issues
Specification	Complex rights and access management, especially with external teams results in loss of data
	Loss of information when tools are managed externally, and no information handover takes place after projects end
	No tight and continuous integration of software users into the specification process of new software features leads to the development of less relevant features
	…
Implementation	Poor tracking or documentation of testing leading to difficulties in deriving new requirements
	Hard to assess and compare quality due to lack of standardised testing and test metrics
	On-boarding of new developers hard due to heterogeneous code base
	…
Integration and Deployment	No focus on user-centered acceptance tests (usability of the UI, clarity of the information presented) but only on functional tests resulting in usability issues for the end user
	Deployment to production environment directly, without intermediate integration testing causing the potential for failures of critical systems
	Pipelines are not available for all repositories and require manual triggering and configuration of the target location making the process error prone and causing a high manual effort
	…
Operation	No error or performance monitoring making it hard to assess the adequacy and functionality of rented resources
	Ad hoc management of errors identified by users and operators leading to inconsistent handling of errors and unclear responsibilities
	No automation within the error handling system causing high manual effort
	…

In summary, it can be stated that initial DevOps approaches were already in place in the company (e.g., SCRUM-based approach, initial pipelines), but implementations were still immature and led to the problem areas mentioned. In addition, different styles of collaboration with external development teams lead to challenges in the implementation of industry-typical collaboration models and in the management of tools. This results in inconsistent and erroneous information flows and friction losses during collaborative development. Overall, there is a great need to increase the speed of development cycles and to ensure the quality of the software addressing these issues.

4.1.2 Synthesis of an industry-specific requirements list for the To-Be DevOps concept

In addition to the analysis of the current state, best practices for DevOps were also reviewed from the literature. The results of the literature review are not presented here in detail, but instead the section focuses only on the comparison with the requirements of the AS-IS situation. In-depth analyses of DevOps best practices can be found, among others, in (Justin Onyarin, 2022). The comparison result is a synthesis of an industry-company-specific requirements list, whose individual requirements were

https://doi.org/10.1017/pds.2023.148 Published online by Cambridge University Press

prioritized in the research project according to the methods described above, and thus represent the implementation goals for the To-Be DevOps concept.
An overview of the synthesized requirements, as well as their origin (requirement from AS-IS situation or best practice) is shown in Table 2. In subsequent discussions, the prioritization of the requirements for the TARGET concept was determined, since not all requirements could be served within the scope of the research project.

Table 2: Overview of Requirements for the implementation of DevOps with their respective priority rank and origins (from the As-Is Analysis [AsIs] or Best Practices [BP] from the Literature)

Rank	Requirements Name	Prioritisation Rank	AsIs	BP
1	Pipelines	Automated pipelines with integrated tests are to be developed.	X	X
2	Version Management	A versioning control tool, such as Git, should be used.	X	X
3	Agile Approach	Use of an agile process model considering company-specific rules.	X	X
4	Testing Strategy	Design of a test strategy whose fulfilment can be verified with metrics and targets.	X	X
5	Monitoring	Monitoring should be established to monitor the progress of a process and analyse whether KPIs and performance thresholds are being met.	X	X
6	Containerized Software	Isolation of the software is to be used for deployment and development of the code (e.g. Docker container).	X	X
7	Different Staging Environments	Use of multiple staging environments adapted to the operating environment.	X	X
8	Incorporation of corporate policies	Consideration of company-specific requirements (Documentation, Collaboration, Access Management, use of open source license etc.)	X	
9	Use of standards	Introduction of standardized procedures for communication, documentation and development.	X	X
10	Control over Tool Administration	The self-administration of IT tools should be able to be administered independently.	X	X
11	Reduced Deployment Effort	The effort required to provide the software should not be a termination criterion.		X
12	Continuous Deployments	There should be automated or manually scheduled, regular deployments in order to test for new security updates.	X	X
13	Adaptability to new Infrastructure	Willingness and acceptance of the team to deal with and work with the new technologies and adapt to new way of working.	X	
14	External Testing	Computationally intensive tests should not be processed in the cloud.		X

The majority of the needs from the AS-IS analysis corresponds to the potential benefits of best practices in the literature. However, the literature pays less attention to the company-specific needs of industrial companies in particular. In addition, common best practice descriptions take less account of the challenges that employees face when converting to new IT tools and process models. This is often a major obstacle for industrial companies with well-established procedures.
By contrast, the need to reduce deployment effort was not identified as a need in the AS-IS analysis. This could change with increasing development and deployment activities in the future. In addition, no need has yet been formulated for compute-intensive testing, which is also frequently named in the literature as an essential best practice for DevOps. Since testing is a major weak point of the As-Is situation anyway, it is plausible that more mature testing procedures have not been named as a current need.

https://doi.org/10.1017/pds.2023.148 Published online by Cambridge University Press

4.2 Concept for To-Be DevOps environment

Based on the requirements and with the help of current best practices, a DevOps concept was created for the production environment as shown in Figure 1. In the BPMN diagram, pools describe entire organizational units, while Lanes indicate who or what performs specific tasks. The pools are the Factory and the Software Department. The actual software development process is triggered by the need for a new feature in the production team and taken up by the software engineering team specifying the need as a requirement. These requirements are then broken down according to SCRUM into tasks that are processed by individual teams of developers writing the respective software code. If the code is in the build stage and is pushed to the appropriate repository, the pipeline simply runs through and deploys the application as a container on a corresponding development environment. If a feature is considered worthy of being rolled out, it first runs through the integration pipeline in order to rigorously test whether it can be rolled out to the production environment without errors. The integration environment therefore has the same specifications (e.g. regarding memory, workload) as the production environment. In addition, all tests must now be passed (e.g. regarding response times) and user acceptance tests must also be completed successfully. As soon as the automatic and manual tests have been completed without any errors, the integration environment is pushed and the performance is monitored continuously. The push can then be made via the production pipeline to the production environment. No more errors should occur in the Production Pipeline after prior, successful deployment to the Integration Environment.

Once the feature has been successfully deployed as a container on the production environment, it is available for use by the production team as well as other software services and can e.g. accesses the machine data via other micro services in the production environment. Once the software is operational, feedback can be sent back to design in the form of direct user feedback or monitored usage metrics via standardised communication channels. If the build/deployment on the production pipeline fails, a roll-back feature handles the error and deploys the "older" version of the software in order to guarantee an error-free application.

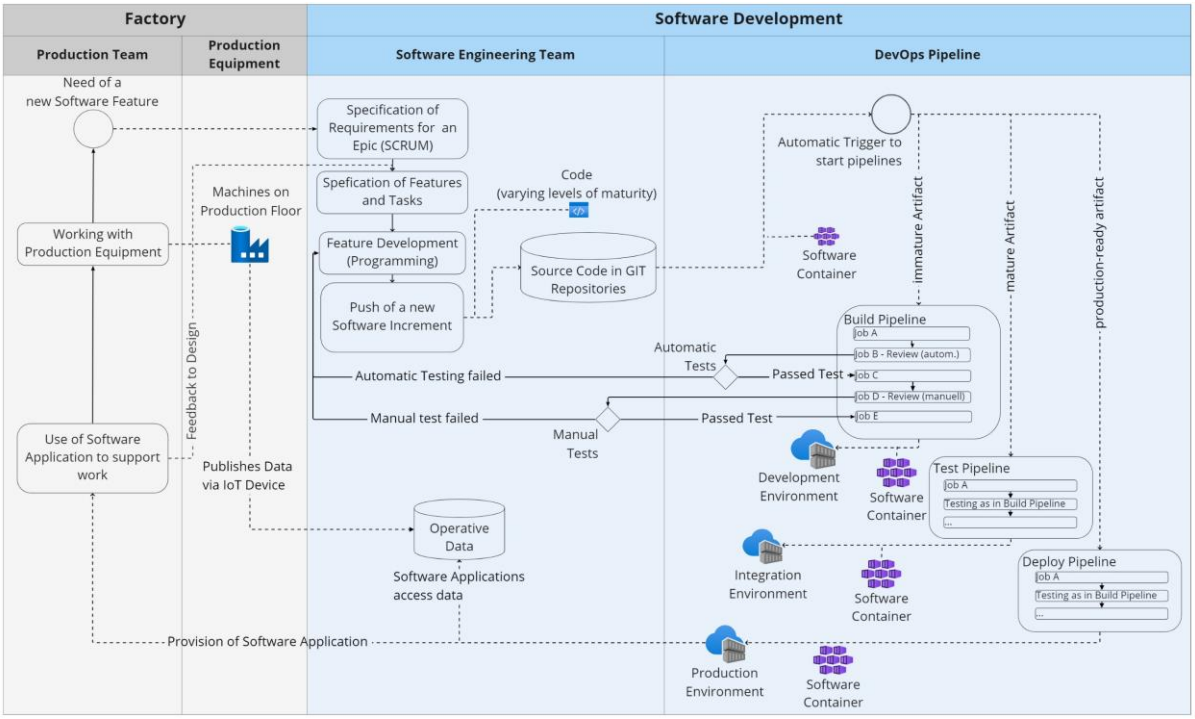

Figure 1: DevOps concept for industrial production environment

The DevOps concept addresses the requirements defined above in short as follows:

Pipelines: Automation of the deployment process occurs, reducing manual work and relying on distinct pipelines depending on the maturity of the software code. A pipeline now exists for each deployable software artefact. All pipelines are identical in the structure of the jobs and the scripts they execute and only differ in terms of target environments to which the software code is deployed and the rigidity with which tests must be passed - both according to the respective maturity level of the

https://doi.org/10.1017/pds.2023.148 Published online by Cambridge University Press

software code. The pipelines (development, integration, production stage) are triggered automatically after each code commit, depending on which development repository is pushed to.
Version Management: The previous version management with Git has been integrated for all pipelines. Automatic rollbacks are possible.
Agile Approach: The SCRUM model used is more effectively supported by the DevOps environment now. Users can provide feedback more quickly, for example, through a ticket system, and are more closely integrated into the development process. There is a high level of transparency for all stakeholders regarding the status of feature development due to a maintained backlog system that includes commenting. Pre-defined roles and responsibility support the team-structure.
Testing Strategy: Metrics and automatic as well as manual tests have been defined, which are used in particular in the integration and production pipeline. Manual approvals are performed by the software engineering team and may also be performed by end users (e.g., user acceptance tests). Error Logs and metrics support the identification of errors and their causes.
Monitoring: The performance of the different environments is monitored (e.g. CPU utilization, communication latency between services, provisioning time). The services of the cloud provider provide various dashboard and reporting functionalities to monitor different performance indicators such as app service state, throughput, utilization, user interactions and consumed resources and their costs. The monitoring also implements an alerting functionality that sends messages to the appropriate responsible parties when certain thresholds are exceeded. An error analysis of the features is possible during the deployment in the pipeline and the operation in the respective environment through detailed error logs.
Containerized Software: The isolation of the software as a container was already evident in the AS-IS situation. In combination with the pipelines, their incremental development and deployment of modular software features has improved.
Different Staging Environments: There are now three staging environments (build, integration, production) via which software can be tested for its functionality as an individual container and as an orchestrated service with other containers, depending on its maturity. The integration of three system environments resulted in increased deployment time while reducing deployment effort. At the same time, manual release processes and acceptance testing of the system environments increase the quality of the software product due to the intensive automated and manual testing. Despite the increased time it takes to deliver the software to the production environment, fewer manual steps are required to deploy the software, significantly reducing deployment effort.
Use of standards: Design Guidelines for programming are defined helping to harmonize the code base. Documentation is improved by implementing different views on code (e.g. user, developer) and predefined authoring tools (IDEs) to match system settings, communication channels and apply the same plugins and pre-sets.
Control over Tool Administration: All IT tools for development and operation (from development tools to software accessing machine on the production floor), are now fully under the company`s own control. This allows greater flexibility in the configuration of individual tools and integration across the tool landscape, e.g. with regard to scaling or access to data and information or implementing standards.
Reduced Deployment Effort: The manual deployment effort is now reduced due automation in the deployment process and standardized communication and programming guidelines. The deployment effort consists of the manual release processes and the actual programming effort of the features.
Continuous Deployments: In order to maintain a secure and updated application, it is necessary to validate package versions and to update them if necessary via a new deployment process. This is achieved through scheduled triggering of specific pipelines.
Adaptability to new Infrastructure: Due to the agile, young and technology-based team, the cultural changes has been adopted without any challenges. The perceived advantages using DevOps principles, methods and tools were evident to the team without further training.
External Testing: The usage of external testing has been validated, but not yet implemented in production due to the currently small size amount of computation needed for the test-set in the deployment pipelines.

4.3 Evaluation of the DevOps concept

As described above, the concept was reviewed by experts and evaluated in comparison to the As-Is situation with regard to the fulfilment of requirements (Figure 2). It is noticeable that significant improvements have been made in almost all areas. The team assessed its adaptability to new

https://doi.org/10.1017/pds.2023.148 Published online by Cambridge University Press

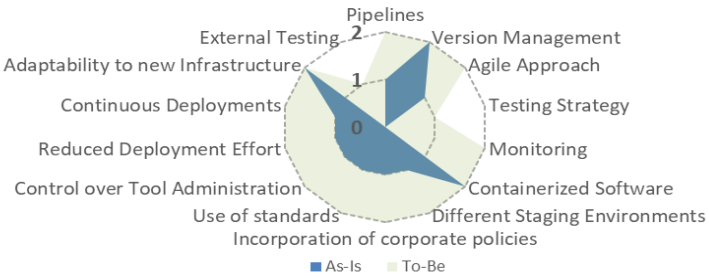

Figure 1: Comparison of As-Is and To-Be situation with regard to addressing the requirements (0-not fulfilled, 1-partly fulfilled, 2-fully fulfilled)

infrastructures as already fully in place, as did the presence of containerized software and version management. In all other areas, the target concept was able to produce improvements. External testing has not yet been fully implemented, which is why the Testing Strategy area is also not yet considered fully established. After the implementation of the DevOps concept, it was evident that the deployment time decreased by 90%. This value was determined by comparing the deployment time for a small change in a software function before and after the introduction of the DevOps concept. It was achieved mainly due to the automatic reduction of complex configuration steps that are now performed automatically. This reduction is independent of the complexity of the software since it relates to the deployment steps which are the same for all changes. Improvements in the quality of software artefacts, e.g., by the test strategy and methodological improvements, must still be observed over time using corresponding KPIs in order to be able to make robust assessments.

5 CONCLUSIONS

One limitation of the case-based research described is that it naturally relates to a single case. Even if production environments in Germany and worldwide are comparable in certain dimensions, no general generalization can be made about the benefits of and ability to implement DevOps in all industrial production contexts. In addition, not all identified DevOps requirements could be fully implemented (e.g., external testing), which means that statements on individual aspects are still incomplete. Another important limitation is that DevOps was only considered here for server-based software artefacts and not for embedded software. Particularly in the product domain and production environment, software that runs directly on machines and interacts with them as well as their environment is becoming increasingly important. In future research, the limitations described should be further explored and a particular focus placed on DevOps for embedded software, since software for an adaptive factory will be called for in the future. It is important to integrate DevOps-based software development with the classic development of (production) systems, e.g. according to the V-model. This includes in particular the modelling of the digital factory system in order to validate software before deployment in the production environment. This becomes especially important when software drops also intervene in the system in a controlling manner. Similar to modern vehicles, new capabilities could then be provided frequently over-the-air in a software-defined manner. It will also be important to be able to absorb industry-specific domain knowledge and integrate it quickly into software and data sets (Preidel and Stark, 2021). Short development cycles and a high degree of collaboration between software developers and domain experts are particularly important for this and their implementation in DevOps should be investigated further.

REFERENCES

Abrahamsson, P., Salo, O., Ronkainen, J. and Warsta, J. (2017), *Agile Software Development Methods: Review and Analysis*, arXiv.

Airaj, M. (2017), "Enable cloud DevOps approach for industry and higher education", *Concurrency and Computation: Practice and Experience*, Vol. 29 No. 5, e3937.

Bass, L., Weber, I.M. and Zhu, L. (2015), *DevOps: A software architect's perspective, The SEI series in software engineering*, Addison-Wesley, Old Tappan, New Jersey.

Dragoni, N., Giallorenzo, S., Lafuente, A.L., Mazzara, M., Montesi, F., Mustafin, R. and Safina, L. (2017), "Microservices: Yesterday, Today, and Tomorrow", in Mazzara, M. and Meyer, B. (Eds.), *Present and Ulterior Software Engineering*, Springer International Publishing, Cham, pp. 195–216.

https://doi.org/10.1017/pds.2023.148 Published online by Cambridge University Press

Dyck, A., Penners, R. and Lichter, H. (2015), "Towards Definitions for Release Engineering and DevOps", in 2015 *IEEE/ACM 3rd International Workshop on Release Engineering, 19.05.2015 - 19.05.2015, Florence, Italy*, IEEE, p. 3.
Ebert, C., Gallardo, G., Hernantes, J. and Serrano, N. (2016), "DevOps", *IEEE Software*, Vol. 33 No. 3, pp. 94–100.
Fisch, Häussler, Can and Diedrich (2018), "Aufbau einer Schnittstelle für die dienstorientierte Automatisierung mittels Analytik-Entscheidungen. Paramterierung von Steuerungen im Cloud-Zeitalter", *apt edition*, 4-5, pp. 26–29.
Fowler (2001), *The agile manifesto*.
Gokarna, M. and Singh, R. (2021), "DevOps: A Historical Review and Future Works", in 2021 *International Conference on Computing, Communication, and Intelligent Systems (ICCCIS), 19.02.2021 - 20.02.2021, Greater Noida, India*, IEEE, pp. 366–371.
Gustavo Schmitz Albino (2022), *Development of a DevOps infrastructure to enhance the deployment of machine learning applications for manufacturing*.
Gustavsson, T. (2016), "Benefits of Agile Project Management in a Non-Software Development Context A Literature Review", in Latvijas Universitate, pp. 114–124.
Hasselbring, W., Henning, S., Latte, B., Möbius, A., Richter, T., Schalk, S. and Wojcieszak, M. (2019), "Industrial DevOps", Vol. 14, pp. 123–126.
Hasselbring, W. and Steinacker, G. (2017), "Microservice Architectures for Scalability, Agility and Reliability in E-Commerce", in 2017 *IEEE International Conference on Software Architecture Workshops (ICSAW), 05.04.2017 - 07.04.2017, Gothenburg, Sweden*, IEEE, pp. 243–246.
Justin Onyarin, O. (2022), "A Complete Guide to DevOps Best Practices".
Kersten, M. (2018), "A Cambrian Explosion of DevOps Tools", *IEEE Software*, Vol. 35 No. 2.
Laukkarinen, T., Kuusinen, K. and Mikkonen, T. (2017), "DevOps in Regulated Software Development: Case Medical Devices", in 2017 *IEEE/ACM 39th International Conference on Software Engineering: New Ideas and Emerging Technologies Results Track (ICSE-NIER), 20.05.2017 - 28.05.2017, Buenos Aires, Argentina*, IEEE, pp. 15–18.
Leite, L., Rocha, C., Kon, F., Milojicic, D. and Meirelles, P. (2020), "A Survey of DevOps Concepts and Challenges", *ACM Computing Surveys*, Vol. 52 No. 6, pp. 1–35.
Lwakatare, L.E., Karvonen, T., Sauvola, T., Kuvaja, P., Olsson, H.H., Bosch, J. and Oivo, M. (2016), "Towards DevOps in the Embedded Systems Domain: Why is It So Hard?", in 2016 *49th Hawaii International Conference on System Sciences (HICSS), 05.01.2016 - 08.01.2016, Koloa, HI*, IEEE, pp. 5437–5446.
Lwakatare, L.E., Kilamo, T., Karvonen, T., Sauvola, T., Heikkilä, V., Itkonen, J., Kuvaja, P., Mikkonen, T., Oivo, M. and Lassenius, C. (2019), "DevOps in practice: A multiple case study of five companies", *Information and Software Technology*, Vol. 114, pp. 217–230.
Lwakatare, L.E., Kuvaja, P. and Oivo, M. (2015), "Dimensions of DevOps", in Lassenius, C., Dingsøyr, T. and Paasivaara, M. (Eds.), *Agile Processes in Software Engineering and Extreme Programming*, *Lecture Notes in Business Information Processing*, Vol. 212, Springer International Publishing, Cham, pp. 212–217.
Macarthy, R.W. and Bass, J.M. (2020), "An Empirical Taxonomy of DevOps in Practice", in 2020 *46th Euromicro Conference on Software Engineering and Advanced Applications (SEAA), 8/26/2020 - 8/28/2020, Portoroz, Slovenia*, IEEE, [S.l.], pp. 221–228.
Preidel, M. and Stark, R. (2021), "SemDaServ: A Systematic Approach for Semantic Data Specification of AI-Based Smart Service Systems", *Applied Sciences*, Vol. 11 No. 11, p. 5148.
Riungu-Kalliosaari, L., Mäkinen, S., Lwakatare, L.E., Tiihonen, J. and Männistö, T. (2016), "DevOps Adoption Benefits and Challenges in Practice: A Case Study", in Abrahamsson, P., Jedlitschka, A., Nguyen Duc, A., Felderer, M., Amasaki, S. and Mikkonen, T. (Eds.), *Product-Focused Software Process Improvement*, *Lecture Notes in Computer Science*, Vol. 10027, Springer International Publishing, Cham, pp. 590–597.
Roy, R., Stark, R., Tracht, K., Takata, S. and Mori, M. (2016), "Continuous maintenance and the future – Foundations and technological challenges", *CIRP Annals*, Vol. 65 No. 2, pp. 667–688.
Rufino, J., Alam, M., Ferreira, J., Rehman, A. and Tsang, K.F. (2017), "Orchestration of containerized microservices for IIoT using Docker", in 2017 *IEEE International Conference on Industrial Technology (ICIT), 22.03.2017 - 25.03.2017, Toronto, ON*, IEEE, pp. 1532–1536.
Ståhl and Bosch (2013), "Experienced benefits of continuous integration in industry software product development: A case study", *The 12th IASTED International Conference on Software Engineering*, Vol. 12, pp. 736–743.
Stark, R. (2021), *Virtual Product Creation in Industry: The Difficult Transformation from IT Enabler Technology to Core Engineering Competence*, 1st edition 2022, Springer Berlin; Springer, Berlin.
Wijaya, P.E., Rosyadi, I. and Taryana, A. (2019), "An attempt to adopt DevOps on embedded system development: empirical evidence", *Journal of Physics: Conference Series*, Vol. 1367 No. 1, p. 12078.
Zimmermann, O. (2017), "Microservices tenets", *Computer Science - Research and Development*, Vol. 32 No. 3-4, pp. 301–310.

https://doi.org/10.1017/pds.2023.148 Published online by Cambridge University Press