Simplifying the DevOps Adoption Process

Ineta Bucena, Marite Kirikova

Riga Technical University, 1 Kalku Street, Riga, LV-1658
ineta.bucena@gmail.com; marite.kirikova@rtu.lv

Abstract. DevOps is a new tendency in business and information technology alignment. The purpose of DevOps is bridging the gap between the development and operations. Several sources claim that DevOps is a new style of work. Many successful DevOps introduction attempts and also many problems in adoption of this style of work have been discussed. This paper reports on research results in facilitating the adoption of DevOps in small enterprises. The DevOps adoption method and several related to it artefacts are proposed. The proposed method has been tested in a national branch of an international company with an internal IT development team.

Keywords: DevOps, agile, maturity model, DevOps practices, DevOps tools

1 Introduction

Today, the business environment is very dynamic with rapid changes in different areas including IT. These rapid changes impact software development process, because customers have high expectations with respect to the content of applications and are demanding higher quality and shorter delivery times [1]. This is one of the reasons why agile practices are widely used and are recommended to be implemented by software development teams.

Still, usually, agile practices are used only on the development side; however, a software development process is not finished with the last acceptance-test and passing the whole deployment over to operations team [2]. Customers cannot use software which is not delivered to the production environment, – and thus the business side cannot get value from the software even if it has been developed very fast. Based on Hüttermann [2], goals of the operations and development teams are different, part of them are even opposite to each other; for instance, the fast change delivery for the development team and the system stability for the operations team. This contradiction does not allow both teams working as one whole. According to reports [3, 4], the “deliver software faster” and the necessity for wider collaboration among IT teams are some of the drivers for adoption of DevOps. Such practices as useful and meaningful collaboration, shared ways of working, or shared goals and trust can lead to better collaboration and understanding between the teams and allow bringing agility into the maintenance part of software development process, too.

The other driving force for DevOps adoption, mentioned in reports [3, 4], is the use of less resources for software development and maintenance, which can be achieved through automation. It is the core activity in one of the DevOps practices - the continuous delivery [5]. DevOps adoption can help to solve the above mentioned and other problems; however, the adoption of DevOps is not trivial and can require complex changes in an enterprise process organization and workflows. To succeed in the DevOps adoption, the enterprises should understand different aspects related to the DevOps approach, have a clear strategy and a plan covering all relevant aspects. They should start the adoption process with the clear idea – what actions should be performed, how they should be prioritized, what tools could support these actions and how to measure the success of the adoption process. Therefore, in this paper, a dedicated DevOps adoption method is proposed for simplifying DevOps adoption in small enterprises. The method could help to save time of enterprises in DevOps adoption and be as a spring-board in the DevOps adoption process.

In order to conceptualize the DevOps adoption process and develop the DevOps adoption method the main research question – "What is the method for simplifying the adoption of DevOps?" was divided in several sub-questions including such questions as "What are the drivers of the DevOps adoption", "What are challenges related to the DevOps adoption?", and 'What are the existing methods regarding to DevOps adoption?". All sub-questions were answered with the help of systematic literature review thus providing the background for the DevOps adoption method's development.

The paper is organized as follows: Section 2 introduces the challenges in DevOps adoption. In Section 3 the proposed method and some of its artifacts are presented. Section 4 reports on an experimental use of the method. Brief conclusions are provided in Section 5.

2 Challenges of DevOps Adoption

Knowledge of potential challenges, derived from available experiences of DevOps adoption, can allow avoiding problems during the adoption process. Even if it is not possible to avoid the problems, this knowledge can help to be more prepared for meeting the challenges. In some cases, the identification of potential challenges can even stop planned adoption processes, if the enterprise decides that it is not possible to overcome the challenges with a reasonable amount of resources.

According to Hamunen [6] DevOps adoption challenges can be grouped in four groups:

- Lack of awareness
- Lack of support
- Problems linked to the DevOps technological implementation
- Problems with adapting organizational processes to DevOps

In this section these four groups of challenges are further used for organized amalgamation of detected DevOps challenges.

Regarding the *lack of awareness in DevOps* the following challenges can be discovered:

- Missing maturity of the concept – no clear definition of DevOps and its practices, no any clear goals available, and a lack of understanding about development workflow phases and responsibilities [3, 6, 7].
- Buzzword perception/ allergy – many people perceive DevOps as a new buzzword, which is just the "last shriek of fashion" and will disappear as soon as has arisen, which does not allow perceiving this approach seriously [6, 7].
- Lack of awareness – which includes missing communication, misinterpretation, insufficient knowledge and a lack of proper training [6, 7, 8].

The following three types of *lack of support for DevOps* adoption process can be identified:

- Lack of management support – including difficulties to "buy-in" senior level management, which can lead to the lack of budget for DevOps adoption [3, 4, 6, 7].
- Lack of team level support (or resistance) – can include either operations team resistance to changes or even development team resistance to changes because of the above described lack of awareness of DevOps [4, 6, 7, 9].
- Lack of trust – including the lack of trust in DevOps idea and lack of trust in people, who promote and work on DevOps adoption process, because of the lack of understanding or missing/ insufficient communication. It can also be caused by just a fear of changes, a fear of potential failures, and a fear of measurements, which could indicate some unpleasant points [3, 6, 7].

Regarding challenges linked to *DevOps technology implementation* the following points can be explored:

- Heterogeneous environment – including test environment limitations due to data management challenges, complexity due to multiple production environments and monolithic system architecture, which should be designed anew to support *DevOps* approach [4, 6, 7, 8].
- Industry constraints and feasibility – because of legacy issues (not having access to production, etc.), compliance concerns, or security and sensitive data issues [3, 4, 7, 8, 9].
- Application types – which is based on the situation that infrastructure does not support modern approaches and does not allow automatic deployments and integration of different tools [4, 6, 7].
- Automated testing – based on worries about test script writing process integration in the code writing process and a support of it [6, 7].
- Tool challenges – includes worries on how to choose the right tools and integrate them with existing tools and practices [6, 7].
- Right scope for monitoring – based on worries about the amount of processes, data, applications, etc., which should be monitored to gather sufficient information for process support and potential improvements, but without putting big weight on the systems [6].

The following challenges with *adapting organizational process to DevOps* can be identified:

- Geographical distribution of teams [6, 7].
- Very complex organizational structure – with too many employees involved in software development process and too many interdependencies. If this is combined with deep-seated company culture with high resistance to changes, it can be a deal-breaker against DevOps adoption process [3, 6, 8].
- Different team capacities – for instance, quality assurance and operations team capacity does not allow to manage development team capacity and deliveries; or business side stakeholders, who cannot support such frequent delivery cycles [4, 8].
- Change management process integration – including existing software development method adaptation or introduction of new software development methods, new metrics adoption, and existing business process integration with the DevOps approach [6].
- Right scope choosing – decisions of the scope can lead to either very good success of DevOps adoption process or vice versa; which puts these decisions to the challenges list [6].

Unlike other sources, Menzel [10] has looked on DevOps challenges from another viewpoint and has identified so called dual-challenges related to DevOps. They can be grouped as in-side and out-side challenges.

Out-side challenges are related to environment, which is around DevOps adoption process, and include the following challenges:

- Wall of confusion based on working in silos, where for each silo (department) there are different goals and minimal information flows between them.
- Speed of innovations based on changes, necessary to be implemented in the systems.
- Complexity of existing environments, where non-production environments do not reflect into production environments, which makes difficult to perform fast root-cause analysis.
- Difficult error prevention and diagnosis process.

In-side challenges include:

- Misunderstanding of DevOps approach and benefits that can be gained from it.
- Perceiving DevOps just as tool implementation, without taking into consideration collaboration, communication and other DevOps culture aspects.
- Difficulties to manage current changes and DevOps adoption, because, usually, there is no option to stop business requests for system development and changes.
- Difficulties to choose the right adoption strategy – whether to choose the Big bang or Step-by-Step based (Phased) approach, as each approach has its pluses and minuses.

3 Method for DevOps Adoption

In the proposed DevOps adoption method, the existing DevOps adoption approaches, original DevOps maturity model and Nine impediment categories framework [11] are integrated with an aim to get a prioritized list of DevOps practices and tools to be applied in small enterprises. The constituents of the method will be discussed in more detail hereafter in this section (see Fig. 1).

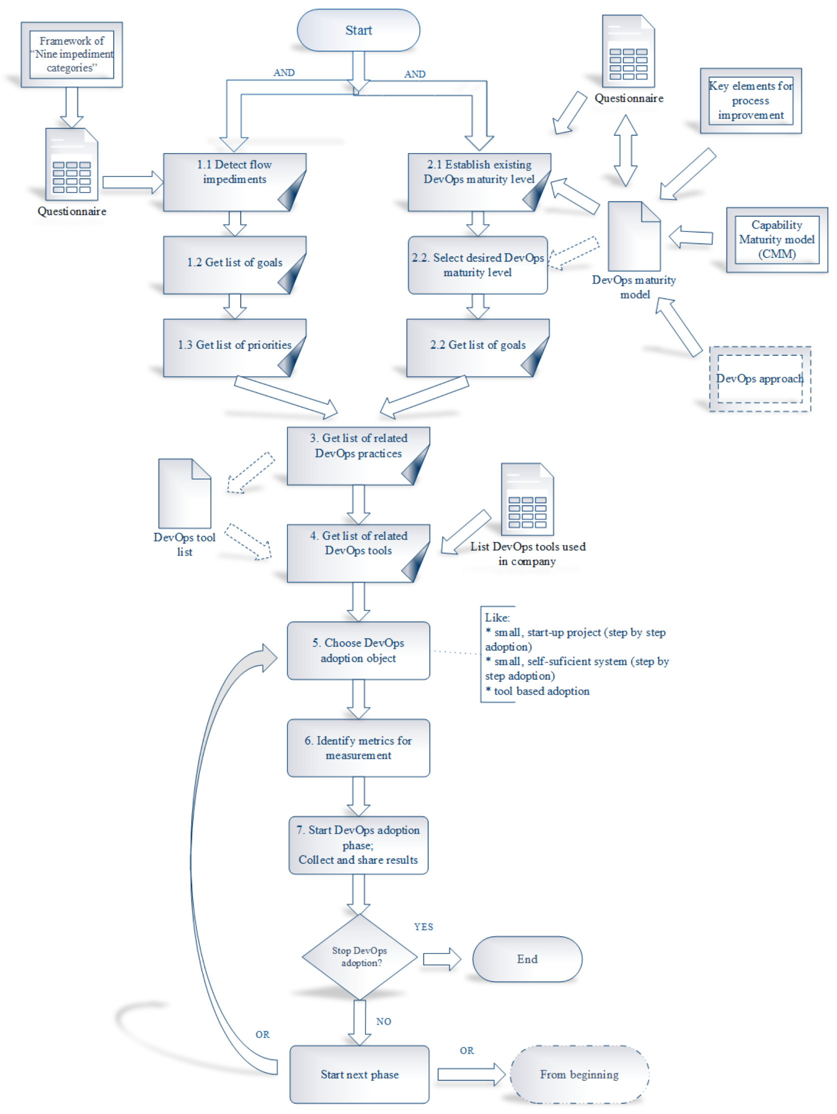
Start
AND
AND
Framework of "Nine impediment categories"
Questionnaire
1.1 Detect flow impediments
1.2 Get list of goals
1.3 Get list of priorities
2.1 Establish existing DevOps maturity level
2.2. Select desired DevOps maturity level
2.2 Get list of goals
Questionnaire
Key elements for process improvement
DevOps maturity model
Capability Maturity model (CMM)
DevOps approach
3. Get list of related DevOps practices
4. Get list of related DevOps tools
DevOps tool list
List DevOps tools used in company
5. Choose DevOps adoption object
Like:
* small, start-up project (step by step adoption)
* small, self-suficient system (step by step adoption)
* tool based adoption
6. Identify metrics for measurement
7. Start DevOps adoption phase; Collect and share results
Stop DevOps adoption?
YES
End
NO
OR
OR
Start next phase
From beginning

Fig. 1. The proposed method for DevOps adoption

The method consists of the following steps, included in the high-level DevOps adoption method's model presented in Fig. 1:

1. Detect impediments to software development flow, based on framework of Nine impediment categories, prioritize them with the help of the questionnaire and get the list of relevant prioritized DevOps practices.
2. Establish the existing DevOps maturity level, based on DevOps Maturity model and select the desired DevOps Maturity level.
3. Get the list of prioritized DevOps practices, by summarizing priorities from the step of impediment identification and relevant DevOps practices, derived from goals, identified in DevOps Maturity model by the help of the identified gap between the existing and desired DevOps Maturity levels.
4. Identify tools existing in organization, related to DevOps approach, and get the list of related DevOps tools, supporting listed DevOps practices.
5. Choose DevOps adoption object for the first phase like a small start-up project, a small self-sufficient system or a tool, which will be a pioneer for DevOps adoption and will bring the most benefit for the team.
6. Identify related metrics for the DevOps adoption object, which will allow measuring the success of adoption process.
7. Process the first phases of DevOps adoption, collect and share results.
8. Start the next DevOps adoption phase by choosing the next DevOps adoption object (step 5) or by starting with step 1 and evaluating again impediments, existing maturity level based on changes that have happened; and set new priorities and related DevOps practices and tools.

Further in this section we will provide information about some of the artefacts mentioned in the description of the steps of the method. Information about all artifacts and guidelines for use of the method are available at the dedicated website at *https://devopsadoptmeth.wordpress.com/*.

The first artifact is the list of impediment categories that, together with the related questionnaire, is used to prioritize the areas where the enterprise wants to improve. "Nine impediment categories" framework includes the following categories [8]:

1. *Work in progress* – it is work, which is not completed and does not provide any value to the customer, who has ordered it. Usually such work is stuck in some of the phases of software development life cycle and is not delivered to production. Similar to "Partially done work" type of waste.
2. *Extra processes* – these are activities that generate extra work, which consumes time and effort without adding value, like additional steps, barriers, documentation or reviews.
3. *Extra features* – development parts, which are added without proven need or validity. They take resources, which could be used for valuable features instead, or create a delay for some value-added features.
4. *Handovers* – activities related to incomplete work handing between persons or groups. Similar to "Task switching" type of waste.

5. *Delays* – related to situations, when one activity should be hold back because of waiting for other activity completion and delivery. Similar to "Waiting" type of waste.
6. *Unnecessary motion* – associated with avoidable movements of people, work or knowledge that create additional inefficiencies or disturb the smooth flow of work.
7. *Failure demand* – refers to the demand on system, team or organization, which is caused by own failure action, or action which is not done. Some similarities with "Defects" type of waste.
8. *Context switching* – related to situations, when people or team should divide their attention for more than one activity at the same time.
9. *Unmet human potential* – related to the waste of not using or fostering people skills and abilities to their full potential.

The next artifact represented here is the DevOps maturity model that was developed on the basis of analysis of related work and includes five levels of maturity with respect to the four enterprise areas, namely, technology, process, people, and culture. The maturity model is represented in Table 1. For each chunk in the maturity model, corresponding DevOps practices reported by DevOps practitioners were associated. Thus the enterprise can see which of these practices are or are not used in the as-is situation of the enterprise. Also the enterprise can see which practices shall be acquired to reach particular maturity levels, and decide on the desired level of maturity, which is realistic for the enterprise's to-be situation. The table that illustrates the practices for the technology area of the maturity model is represented in Table 2. Similar tables are also developed for other maturity model areas.

Table 1. Proposed DevOps Maturity model

Area	**ID**	**Initial level (1)**	**Repeatable level (2)**	**Defined level (3)**	**Managed level (4)**	**Optimized level (5)**
TECHNOLOGY	T1	Environments are provisioned manually	All environment configurat-ions are externalized and versioned	Virtualization used if applicable	All environ-ments managed effectively	Environment provisioning fully automated
	T2	Manual tests or minimal automation	Functional test automation	Triggered automated tests	Smoked tests and dashboard shared with Op.t.	Chaos Monkey
	T3	Data migration un-versioned and performed manually	Changes to DB done with automated scripts versioned with application	DB changes performed automatically as part of deployment process	DB upgrades and rollbacks tested with every deployment	Feedback from DB performance after each release

Area	ID	Initial level (1)	Repeatable level (2)	Defined level (3)	Managed level (4)	Optimized level (5)
	T4	Manual deployment	Build automation	Non-production deployment automation	Production deployment automation	Op.t. and Dev.t. regularly collaborate to manage risks and reduce cycle time
	T5	Manual processes for building software/ No artifact versioning	Regular automated build and testing. Any builds can be recreated from source	Automated build and test cycle every time a change is committed	Build metrics gathered, made visible and taken into account	Continuous work on process improvement, better visibility, faster feedback
	T6	No collaboration tools	Project planning tool	Team/ toolset integration	Knowledge manage-ment tool	–
	T7	No software configuration management (SCM)	Standardized SCM	Configuration is delivered together with code	Self-healing tools	–
	T8	No or minimal monitoring	Core monitoring	Integrated monitoring	Analytics/ Intelligence	–
	T9	No tools or minimal tool usage for issue tracking	All issue and bug reports are tracked	Issue reporting automatization and monitoring	Activities based on received feedback and data	Continuous delivery process
	PR 1	Inconsistent delivery process	Scheduled delivery process	Automated delivery process	Frequent delivery process	Development process integrated with Six sigma
	PR 2	Ad-hoc development	Scrum development	Agile development	Lean develop-ment	Continuous testing
PROCESS	PR 3	Ad-hoc testing	Requirement based testing	Integrated testing	Qualitative testing	Organized performance management
	PR 4	Inconsistent project management	Project & requirement management	Integrated project management	Quantitative proj. manage-ment	–
	PR 5	Deployment and development documenta-	Development documenta-tion and relevant	Regular validation of the documentation	Documentat ion process and structure	–

Area	ID	Initial level (1)	Repeatable level (2)	Defined level (3)	Managed level (4)	Optimized level (5)
		tion is not available or is out of date	configuration files are up-to-date	and related configuration descriptions are provided	update based on gathered experience and quality require-ments	
	PR 6	Uncontrolled or reactive processes (not applied management)	Processes are managed, but not standardized	Processes are standardized across organization	Visibility & predictabili-ty of entire process & performanc e	Highly optimized & integrated processes
PEOPLE	P1	Teams organized around skillsets	Team organized around deliveries	Team organized around projects	Team organized around products/ business lines	Interdisciplin ary teams organized around KPIs
	P2	Ad-hoc learning	Team learning	Value stream learning	X-process learning	External learning
	P3	Ad-hoc approach regarding competences development	Competences are developed with the help of training and development	Analysis of exiting competences and future development	Mentor usage	Continuous capability improvement
CULTURE	C1	Restricted communicati on	Rapid intra-team (inside) communicati on	Rapid communicatio n between teams (inter-team)	Frequent, collaborative communicati on	Rapid feedback
	C2	Uncommunic ated vision	Clear delivery requirements	Clear project requirements	Clear product/ business line require-ments	Clear organization requirements
	C3	Lack of awareness of how culture is impacting day-to-day business	Awareness of aspects in culture that may help or hinder day-to-day business	Cultural traits that support business strategies have been identified	Culture viewed as an asset to be managed	Desired elements of the culture are identified, ingrained and sustainable, thus creating "the way we work here"

Area	ID	Initial level (1)	Repeatable level (2)	Defined level (3)	Managed level (4)	Optimized level (5)
	C4	Poor, ad-hoc communication and coordination	Managed communication	Active collaboration	Collaboration based on process measurement, which allows to identify bottlenecks and inefficiencies	–
	C5	Sub-innovating/ no innovations	Innovations by necessity	Innovation by design	Strategic innovation	–

Labels in Table 1: Empty model cells, in which relevant goals are not provided, are depicted using "–", Opt.t – operations team, Dev.t. – development team.

For the purpose of better utilization of information represented in the tables, there are specific questionnaires made that help to identify the information according to the developed maturity model and the tables of related practices. They are available at *https://devopsadoptmeth.wordpress.com*.

Table 2. DevOps practices relevant to Maturity Model Technology area

TECHNOLOGY maturity levels				
ID	**Repeatable (2)**	**Defined (3)**	**Managed (4)**	**Optimized (5.)**
T1	*Hardware maintenance practices*	Integrated configuration management		
		Server virtualization practices	▨	Infrastructure as code
T2	Automated testing		Continuous testing	Continuous experimentation and learning
			Continuous monitoring	
			Automated dashboards	
T3	Collaborative development	Integrated Deployment planning	Continuous testing	Continuous monitoring
	DB management practices			
	▨	▨	▨	Application monitoring
T4	Build automation	Continuous integration	Continuous deployment	Integrated change management
				Constant, effortless communication and collaboration
				Active Stakeholder participation

TECHNOLOGY maturity levels				
ID	**Repeatable (2)**	**Defined (3)**	**Managed (4)**	**Optimized (5.)**
T5	Collaborative development	Continuous integration	Common metrics for Dev.t. and Op.t.	Continuous delivery
			Continuous monitoring	Constant, effortless communication and collaboration
T6	Continuous business planning	Continuous integration	*Knowledge management practices*	–
	Integrated Deployment planning		Constant, effortless communication and collaboration	
	▨	Active Stakeholder participation		
T7	Integrated configuration management			–
	▨	Infrastructure as code	Continuous experimentation and learning	
T8	Application monitoring	Continuous customer monitoring and feedback	Automated dashboards	–
	Continuous monitoring			
	Common metrics for Dev.t. and Op.t.			
T9	Production support	Integrated change management	Shared goals, values, respect, trust and incentives	–
	Shared responsibility, ways of working and collective ownership	Integrated Deployment planning		

Tools are an important part of the *DevOps* approach. They are tightly related to different DevOps practices, such as automation, monitoring or integrated configuration, and help to introduce these practices.

Based on seven different sources, namely, [12, 13, 14, 15, 16, 17, 18], eleven toolchain groups were identified. They are shown in the first column of Table 3. A specific table-like form of representation of toolchain groups and tools related to each group was established. This form of representation makes it more comfortable to identify necessary tools for DevOps practices chosen by enterprises. Having knowledge on toolchain groups and tools related to the chosen DevOps practices, the enterprises can see what investments will be needed to ensure the tool support for the desired levels of DevOps maturity. The correspondence between the toolchain groups and DevOps practices is not shown here due to space limits, – it is represented at *https://devopsadoptmeth.wordpress.com*.

Table 3. Tools grouped in DevOps toolchain groups

Toolchain groups	**Related tools**					
Version and source control	Git/ GitHub/ GitLab	Mercurial/ Bitbucket	Subversion			
Containerization	Docker	Mercurial/ Bitbucket	Rocker		Vagrant	
Configur. managem. tools	Docker	Puppet	Ansible	Cheff	Vagrant	SaltStack
Continuous Deployment	Capistrano	Jenkins	Ansible	Codeship	Travis CI	circle CI
Contin. integrat. and orchestr.	Atlassian Bamboo	Jenkins	TeamCity	Codeship	Travis CI	circle CI
Build automation	Apache Maven Proj.	Jenkins	Apache Ant	Gradle	Travis CI	
Automat. Test. and validat.	Cucumber	Selenium	TestComplete	Jmeter		
Monitoring	Zabbix	New Relica	Nagios	Splunk	AppDy namics	
Collaboration tools	Slack	Jira	HipChat	Pager Duty		
Issue tracking	Bugzilla	Jira	Track/ tesTrack	MantisBT	Assembla	
Planning tools	Clarizen	Jira	Confluence	Asana		
Knowledge sharing tools	Crowdbase	Nuclion	Confluence			
DB handling tools	DBMaestro	LiquiBase	RedGate			

4 Experimental Application of the Method

The experimental application of the method was done in Company X that corresponds to small and medium sized enterprise definition as a medium-sized enterprise with ~220–250 employees. Its main business is not related to IT. There are more than 500 outside collaborators, too. If we look at the proportion of number of system users vs. number of IT department employees (19+2) it could be approximately 1:200, because multiple collaborating partners' employees are also using organization's systems. The IT department can be considered as a small enterprise. Company X runs the business line where one of the main drivers is the possibility to react fast on clients' needs and environment's changes. All core business processes are related to the organization's back-end system, which is developed using in-house development resources. The main part of collaboration partners and agents are using organization's front-end web based system, which is linked to the back-end system using middleware services. Company X has a web based self-services system for clients, too, which is linked to the back-end system similarly as collaboration partner systems. All web systems and middleware services are developed using outsourced developers. The similar situation is with organization's web based CRM system.

At this moment, IT department is divided in four silos, where one of them is Maintenance (operations) team and the other three are development teams (shared service centers (SSC)), which are divided based on high level business lines. One of the SSCs is located abroad in a different country. There is a testing team with one quality assurance specialist per SSC and a Web system development process manager.

According to information, which was received during DevOps adoption method’s application process (the company representatives applied firs 6 steps of the method); Company X was already using some of DevOps practices, which had allowed reaching the second DevOps maturity level in Technology area. Still there was not enough progress in other three areas, where maturity level was only at the first level. As all four areas were similarly important for a successful change implementation, it was essential to work on other area’s maturity level rising.

Regarding the prioritized DevOps practices listed, one of the highest priorities for Company X was automated testing practice, which was important to fill the gap relevant to the chosen maturity level and was one of the practices relevant to highly rated software development flow impediments. Other highest level priority practices were collaboration practices (e.g. active stakeholder participation, shared responsibilities, knowledge sharing practices, etc.).

Some attention had to be turned to DevOps practices, which were with high priority level based on impediment identification, but regarding DevOps maturity level were already established in the company. For instance, build automation, which supports second Maturity level and was assigned as a practice, which was already used. Probably there was still high level of waste because the practice was neither fully established yet nor established in all development processes. Similarly it was with constant effortless communication and collaboration.

The application of the method shoved that Company X has already a good tool set, where some of them like Jenkins, Jira, and Confluence were usable for more than one DevOps practice support, which meant that there were space for growth without high additional investment. It was also identified that tools, relevant to DevOps practices, which could be useful for Company X during DevOps adoption process were Containerization tools like Docker, Mercurial/ Bitbucket, Rocker and Vagrant and DB handling tools like DBMaestro, LiquiBase, or RedGate.

During the experiment it was necessary to clarify some issues with the company; and some possible future elaborations helpful in method's application were detected. In overall the method was applied successfully and Company X was able to gain the needed information for decision making regarding DevOps adoption.

The application of the method showed that the proposed method can help to meet (at least partly) all four groups of DevOps adoption challenges discussed in Section 2. It can clarify the DevOps concept. With clear representation of different DevOps maturity levels and supplementary questionnaires, it can facilitate communication regarding management support for introducing new DevOps practices. Concerning technical challenges, it helps to identify new technologies that can be well integrated in enterprise infrastructures. The same applies to adapting organizational processes to DevOps as the method helps to choose the right scope of steps to be performed in

DevOps introduction at chosen levels of maturity. While the method does not address every single challenge mentioned in Section 2, by addressing part of them, it can positively influence enterprise ability to meet other challenges, too.

5 Conclusions

The purpose of this research was to design and validate the DevOps adoption method to guide small enterprises in DevOps adoption process and also to simplify the adoption process.

According to the DevOps adoption concept and relevant information summarized during literature review, which allowed identifying main points leading to the beginning of the DevOps adoption process, the design of the DevOps adoption method was started. The necessity to determine the existing situation in an enterprise and define the desired level of maturity was identified. Therefore the DevOps Maturity model was designed. The model was related to amalgamated DevOps practices and tools. The “Nine impediment categories” framework was used as to identify necessary DevOps practices, which a particular company could use during the DevOps adoption process in order to achieve the desired level of DevOps Maturity. To provide access to the method's description and its supplementary artifacts the web page was created at *https://devopsadoptmeth.wordpress.com/.* This page was used during method validation process described in Section 4 to provide easy access to the developed questionnaires.

The DevOps adoption method validation, performed with the help of a particular company’s IT team showed that the method allows to determine the list of prioritized DevOps practices and allows to derive the list of tools supporting these practices. After choosing the DevOps adoption object and establishing relevant metrics, an enterprise can proceed with the DevOps adoption process with the help of the derived DevOps practices and the identified list of supporting tools. The use of the method can help to meet several challenges of DevOps adoption mentioned in Section 2.

The method provides an opportunity of simplification of DevOps adoption in small enterprises. It has not been analyzed yet whether it is also useful for large enterprises.

The further research concerns application of the method in more companies, the DevOps adoption management tool development, the method's tuning; and the development of the approach for systemic artefact maintenance to ensure that all constituents of the method are continuously up-to date and consistent.

Acknowledgment. This work is supported in part by the Latvian National research program SOPHIS under grant agreement Nr.10-4/VPP-4/11.

References

1. Waters, K.: All about agile. CreateSpace Independent Publishing Platform, pp. 380 (2012).
2. Hüttermann, M.: DevOps for Developers – integrate development and operations, the agile way. Second edition, Apress, pp. 196 (2012).

3. CA Technologies, White paper: What smart businesses know about DevOps, (2014). http://www3.ca.com/~/media/Files/whitepapers/techinsights-report-what-smart-businesses-know-about-devops.pdf, last accessed 2017/08/10
4. Gleanster & Delphic, 2015 Annual State of DevOps, (2016). https://puppet.com/resources/whitepaper/state-of-devops-report?pcnav=off&pctiles=off&ls=Campaigns&lsd=Search&cid=7010f000001eViP&utm_medium=paid-search&utm_campaign=Q2FY18_EMEA_All_CAMPGN_SER_ADWRDS_2016-DO-sal-rpt&utm_source=google&utm_content=devops-salary-report&gclid=Cj0KCQjwn6DMBRC0ARIsAHZtCeO41o1MUzVXRCsTh6SPV2uEyfJTY3FG4mU2WKyTogAM5ffTw1akJJwaAs-JEALw_wcB, last accessed 2017/08/10
5. Sharma, S., Coyne B.: DevOps for Dummies. Second IBM limited edition, USA, John Wiley & Sons, Inc.Hoboken, (2015).
6. Hamunen, J., Challenges in Adoption a DevOps Approach to Software Development and Operations. Master thesis, Aalto Univerity, School of Business, Aalto, Finland (2016).
7. Amaradri. A.S., Nutalapti. S.B.: Continuous Integration, Deployment and Testing in DevOps Environment. Master thesis, Blekinge Institute of Technology, Faculty of computing, Karlskorna, Sweden, pp. 115 (2016).
8. Riungu-Kalliosaari, L., Mäkinen, S., Lwakatare, L.E., Tiihonen, J., Männistö, T.: DevOps Adoption Benefits and Challenges in Practice: A Case Study. In: Product-Focused Software Process Improvement, PROFES 2016, Lecture Notes in Computer Science, vol. 10027., Abrahamsson P., Jedlitschka A., Nguyen Duc A., Felderer M., Amasaki S., Mikkonen T. Ed., Springer, Cham, pp. 590-597 (2016).
9. Jones, S., Noopen, J., Lettice, F.: Management challenges for DevOps Adoption within UK SMEs. In: QUDOS 2016 Proceedings of the 2nd International Workshop on Quality-Aware DevOps, USA, ACM, pp. 7-11 (2016).
10. Menzel, G.: DevOps – Don't be left behind. (2015). https://www.capgemini.com/blog/capping-it-off/2015/08/*DevOps*-dont-be-left-behind, last accessed 2017/08/10
11. Power, K., Conboy, K.: Impediments to flow: rethinking the lean concept of 'waste' in modern software development. In: Agile Processes in Software Engineering and Extreme Programming. XP 2014. Lecture Notes in Business Information Processing, vol 179., Cantone G., Marchesi M. Ed., Springer, Cham, pp. 203–217 (2014).
12. Puppet: How to Build a High-Performing IT Team. [https://puppet.com/resources/whitepaper/how-build-high-performing-it-team, last accessed 2017/08/10
13. Yehuda, Y.: 11 tools you must have in your DevOps toolchain. (2015). http://www.dbmaestro.com/2015/10/infographic-11-tools-you-must-have-in-your-DevOps-toolchain/, last accessed 2017/08/10
14. Akshaya, H.L., Nisarga, J.S., Vidya, J., Veena, K.: A Basic Introduction to DevOps Tools. In: International Journal of computer Science and Information Technologies 6(3), (2015).
15. Upguard, DevOps Toolchain, (2015). https://www.upguard.com/hs-fs/hub/228391/file-2341634679-pdf/DevOps_ScriptRock.pdf, last accessed 2017/08/10
16. XebiaLabs Periodic table of DevOps tools, (2015). https://xebialabs.com/periodic-table-of-DevOps-tools/, last accessed 2017/08/10
17. David Linthicum (2016). http://www.techtarget.com/contributor/David-Linthicum/2016, last accessed 2017/08/10
18. Wilinski E., DevOps Best Practices: Finding the right tools, (2014). https://blog.newrelic.com/2014/06/02/DevOps-tools/, last accessed 2017/08/10