Development of a GQM-based Technique for Assessing DevOps Maturity

Thomas Neubrand[1,2] and Thorsten Haendler[1,2]
[1] *University of Applied Sciences BFI Vienna, Austria*
[2] *Vienna University of Economics and Business (WU), Austria*

Keywords: DevOps, Maturity Assessment, Goal Question Metric (GQM), Continuous Delivery, Survey.

Abstract: DevOps aims to increase an organization's ability to build, test and release software in a rapid, frequent, and more reliable fashion by combining the perspectives of software development and IT operations. It emphasizes the communication and collaboration among all stakeholders involved in the software development lifecycle by applying a set of technological and cultural practices and principals. Organizations are increasingly adopting DevOps practices in order to capitalize on its benefits. However, DevOps is a paradigm with different interpretations and definitions, making it difficult for organizations to decide what practices and capabilities of DevOps to adopt and to enhance based on their individual business needs. In this paper, we propose a GQM-based approach for assessing the DevOps maturity in organizations. For this purpose, we structure DevOps capabilities in terms of goals and sub-goals that enable DevOps identified in existing research literature. Then, questions and metrics are derived and operated for a questionnaire to assess the DevOps maturity with regard to the different capability levels. The resulting maturity report is finally illustrated via different kinds of radar charts representing the individual levels of the identified goals and sub-goals. A first evaluation with experts indicates that the developed technique seems useful to assess DevOps maturity. With the support of the proposed technique, organizations are able to assess its current practices and measure the improvements over the course of time.

1 INTRODUCTION

In the era of digital everything, software is becoming more than ever an essential form of differentiation. Yet many companies struggle to keep pace with the range of new demands. The constant change of business needs, associated with the requirement of a reduced time to market, has changed the speed of application development and shortened the release cycle considerably in the last decade. For many organizations, adopting DevOps has been the answer to these challenges (Farcic, 2019).

DevOps emphasizes the advantages of developing products in a multi-perspective way, so that developers, designers as well as operations, security and business people all have a common understanding of what needs to be accomplished (Bass et al., 2015). It aims to improve the transfer of knowledge between teams and focuses on reducing hand-off time.

However, DevOps is not a simple process, perspective, or tool chain that can be implemented. It rather is a set of coherent cultural and technological principles and practices that can be adopted and improved over time (Forsgren et al., 2014).

Regarding the question what practices to adopt, DevOps leaves willingly room for interpretation, making it difficult for organizations to decide what practices to adopt and when to be considered as fully implemented (Harrison et al., 2019). By frequently assessing the DevOps maturity of an organization, potentials and bottlenecks are made transparent to other areas in the organization and supports the understanding of what is needed for a successful DevOps adoption (Smeds et al., 2015).

In general, maturity models and assessment techniques are popular instruments to provide guidance for improvement activities and can be a useful metric to analyze adoption processes, improvements potentials or demand for resources (van Hillegersberg, 2019). However, in current research literature only a very few approaches for assessing DevOps maturity can be identified. These approaches either focus on specifying maturity levels with little focus on actually assessing the maturity (Bucena and Kirikova, 2017; Radstaak, 2019) or provide assessment in an rather unstructured way (also see Section 2.2), which are both less suited to assess DevOps maturity in organizations.

Neubrand, T. and Haendler, T.
Development of a GQM-based Technique for Assessing DevOps Maturity.
DOI: 10.5220/0010177801170129
In *Proceedings of the 12th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management (IC3K 2020) - Volume 3: KMIS*, pages 117-129
ISBN: 978-989-758-474-9

In this paper, we propose an approach for a survey-based technique for assessing the DevOps maturity level in organizations. In particular, this paper provides the following contributions:

- We propose a GQM-based technique for assessing the DevOps maturity in organizations.
- We present a conceptual model structuring 33 goals and sub-goals enabling DevOps, from which a set of 98 questions is derived for the survey-based assessment.
- We discuss the feedback and a first evaluation of the proposed assessment technique based on an expert survey. In addition, this feedback has been included to improve the model and assessment.

The remainder of this paper is structured as follows. Section 2 reflects on the thematic background and discusses research related to the proposed approach. In Section 3, we describe the applied research design and explain in detail the goals/factors applied for assessing DevOps. Subsequently in Section 4, we introduce the assessment technique and discuss the results of a first evaluation. In Section 5, we reflect on limitations of the approach and describe further potential in terms of exemplary use cases. Finally, Section 6 concludes the paper.

2 BACKGROUND AND RELATED WORK

In this section, we provide background information on the purpose and techniques of DevOps adoption and assessment (Section 2.1) and discuss research related to our approach (Section 2.2).

2.1 DevOps Adoption and Assessment

The journey of adopting DevOps in organizations brings with it many challenges, both technological and cultural. DevOps allows for various trends and ideas that can be associated with it, which raises the question of how DevOps can be adopted and further on assessed without a ubiquitous definition or a defined set of practices (van Ommeren et al., 2016). A clear business objective supports the transformation by prioritizing the work along the way and thus maximizing the impact. Since it is impossible to know all the right decisions on an enterprise-level from the very beginning, a rigorous feedback loop has to be applied, in order to make the appropriate adjustments in each iteration (Gruver et al., 2015). For this purpose, in order to identify potential fields and aspects that require attention, the current state of DevOps in the organization needs to be investigated. The proposed assessment technique can provide guidance in deciding what DevOps practice to improve and where to invest further resources based on the perspectives and estimations of the involved stakeholders.

2.2 Related Work

Research related to our approach can be roughly divided into (1) maturity models and (2) assessment techniques, which are both discussed below.

Maturity Models. In the domains of Information Systems and Software Engineering, the application of maturity models is quite popular (Wendler, 2012; Mettler, 2011; Becker et al., 2010; Poeppelbuss et al., 2011; von Wangenheim et al., 2010). Most of these maturity models represent variants of the capability maturity model (CMM) and capability maturity model integration (CMMI) approach (Lasrado et al., 2015). These models have the advantage that they display the maturity level clearly on a few distinguishable levels, e.g. from initial (0) via defined (3) to optimized (5). A disadvantage is that details of the individual factors are not presented. Another type of maturity model are focus-area maturity models, which consist of different focus areas that need to be addressed in order to achieve maturity in a functional domain (van Hillegersberg, 2019). However, for DevOps in particular, only a few maturity models can be identified in current research literature. For instance, a guidance in the DevOps adoption process in the format of a maturity model is described by (Bucena and Kirikova, 2017) and (Radstaak, 2019). These approaches have in common that they focus on different dimensions of DevOps including areas such as technology, process, people, and culture, which is similar to our approach. In addition to specifying maturity levels, our approach provides means to actually assess and report the DevOps maturity for a set of relevant capabilities.

Adoption and Assessment Techniques. For decades, we see efforts in Software Engineering and Information Systems to measure the state of software artifacts and processes especially in order to identify shortcomings that can be addressed through targeted remedial activities. For this purpose several metrics are established (Fenton and Bieman, 2014). A general problem with metrics is that often just those aspects are measured, which are easily measurable. DevOps in particular provides multiple challenges, since it represents a paradigm with different interpretations and definitions as well as involving technical, cultural as well as organizational aspects. For this reason, for assessing DevOps maturity only very few assessment approaches can be identified. A prominent example represents the technique provided by *DORA's* research program, which provides a quick check to measure the software delivery performance of teams and provide improvement areas for low performers[1]. In contrast to this approach, our technique offers a more comprehensive analysis by considering a total of 33 sub-factors. Another example is the GQM-based approach presented in (König and Steffens, 2018) that pursues a similar technique. However, we extend the presented approach by covering more aspects impacting DevOps maturity and by providing means to actually assess and report the DevOps maturity level.

3 DEVOPS MATURITY MODEL

In this section, we describe the applied research design (Section 3.1) and introduce the concept model for assessing DevOps maturity (Section 3.2).

3.1 Research Design

In order to address the aforementioned challenges, this paper aims to investigate the development of a lightweight and flexible technique for assessing DevOps maturity in organizations. From a research perspective, the approach is oriented to design-science research for developing and evaluating the assessment technique. Moreover, it leverages a Goal Question Metric approach for correlating goals and metrics for assessing DevOps maturity.

Design-science Research. The development of the maturity model and assessment technique is guided by a design-science research process (Hevner et al., 2004; Peffers et al., 2007). Key to this research approach is to build and evaluate artifacts that aim at solving problems considered relevant. If the evaluation reveals points for improvement or extension, further iterations can be performed. This approach is well suited for developing and evaluating our assessment technique, since the DevOps paradigm is not clearly defined and dynamic (e.g. with regard to the scope of the practices/capabilities?), which might require adjustment over time. The resulting model and assessment technique has been created in multiple iterations of designing and evaluating. However, just as DevOps is based on the idea of continuous improvement, this model can be seen as a "*living artifact*", which is improved and adapted over the course of time.

[1]https://cloud.google.com/devops

Goal Question Metric. We leverage the Goal Question Metric (GQM) approach to measure the DevOps maturity of organizations (Caldiera and Rombach, 1994). GQM represents a structured top-down approach, starting by defining the goals and sub-goals. Then a set of questions with metrics to characterize and assess the specific goals is derived; also see Fig. 1.

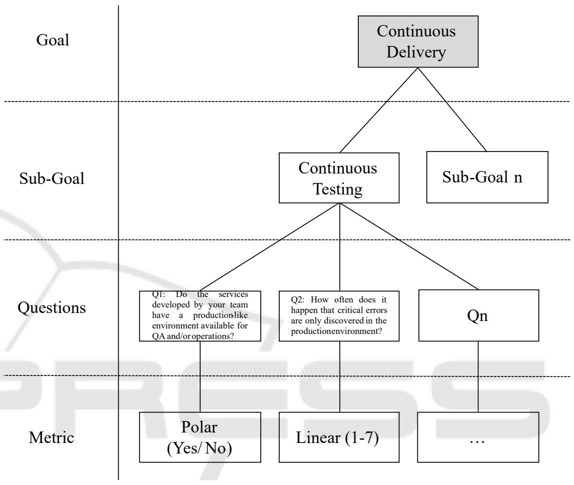
Goal
Continuous Delivery
Sub-Goal
Continuous Testing
Sub-Goal n
Questions
Q1: Do the services developed by your team have a production-like environment available for QA and/or operations?
Q2: How often does it happen that critical errors are only discovered in the productionenvironment?
Qn
Metric
Polar (Yes/ No)
Linear (1-7)
…

Figure 1: GQM applied to assessing DevOps maturity (exemplary excerpt). For further details, also see Tables 1 and 2 and Sections 3.1 and 3.2.

For specifying the goals and sub-goals, we orient to the research presented in the yearly *State of DevOps* reports conducted by the *DevOps Research and Assessment (DORA)* team (Forsgren et al., 2019). *DORA* is an ongoing research program starting in 2014, which applies rigorous methodology to analyze DevOps capabilities and provides support regarding what technical, process, measurement, and cultural practices need to be adopted to achieve higher software delivery and organizational performance (Forsgren et al., 2019). Accordingly, we were guided by their research, based on which the following five groups of practices can be identified that contribute to the DevOps adoption process:

- Lean Management
- Culture and Work Environment
- Transformational Leadership
- Lean Product Management
- Continuous Delivery

Based on these goals, sub-goals have been structured and aggregated. Fig. 2 illustrates goals and sub-goals in terms of a conceptual goal model (Caldiera and Rombach, 1994). For specifying questions for the survey-based assessment, we derived questions from each sub-goal based on research literature (see Tables 1 and 2). Survey-based data is particularly well suited at providing a holistic view of systems and can be leveraged to provide reliable insights for organizations. Also, survey-based data can be gathered much faster than system data to start off with, and later put in comparison and eventually find root-causes for deviations (Forsgren et al., 2018). The metrics applied in this approach are subjective evaluations of the participants depending on their point of view (e.g. their role in the software development lifecycle). However, this technique addresses the challenges that objective metrics are especially difficult to implement for cultural, but also for technological practices and principles. Fig. 1 illustrates the structured approach of deriving questions and metrics from DevOps goals utilized in this paper.

3.2 DevOps Capabilities

To develop the technique to assess DevOps maturity, different practices associated with DevOps were examined. As stated in Section 2, the practices related to DevOps are interpreted differently, which leaves room for an individual approach. The annual *DORA* DevOps report provides a solid foundation for analyzing DevOps practices applied in industry that drive high performance software delivery and operational performance (Forsgren et al., 2019). For the purpose of designing the assessment technique, we build on the results of the *State of DevOps* reports and extended them by further concepts identified in research literature.

Fig. 2 depicts the 5 identified key factors enabling DevOps (goals) with aggregated sub-factors (sub-goals). In particular, the concept map is specified as a class diagram of the Unified Modeling Language (UML2) (Object Management Group, 2017). In particular, each factor is represented as a class and the classes are connected via different kinds of associations, e.g. *enable* associations or aggregations. The individual organizational conditions represent the different variable instances of this class diagram. To what extent these conditions can be quantified is discussed in Section 5. Please note that the multiplicities (i.e. number of possible class instances per relation) are not specified in this diagram, since the classes mostly represent more abstract concepts or activities. However, for each factor or aggregated sub-factor a multiplicity of 1 could be applied.

In the following, the concepts and their relevance for DevOps are explained in detail, structured by the 5 goals.

Lean Management. In the 2015 *State of DevOps Report*, the researchers found that Lean Management practices contribute to creating a culture of learning and continuous improvement, lower levels of burnout, and higher organizational performance overall (Forsgren, 2015). They identified the following practices as most significant (also see the classes on the top in Fig. 2).

- `Business Decisions based on Metrics`: Utilizing data from application and infrastructure monitoring tools is key to gain insight into the systems and to guide business decisions on a daily basis. It facilitates a feedback loop from production systems to its responsible team members (Forsgren, 2015).
- `Visual Task Boards`: Visualizing important information to monitor quality, productivity and work in process helps creating a shared understanding between different stakeholders and within the team itself, identify bottlenecks and shows operational effectiveness (Forsgren, 2015).
- `Limiting Work in Progress`: By visualizing Work in Progress (WIP), stakeholders are able to prioritize work in the context of global objectives. WIP limits are preventing bottlenecks as well as constraints from happening and ultimately drive process improvement (Forsgren et al., 2017; Kim et al., 2016).

Transformational Leadership. In the 2017 *State of DevOps Report*, the researchers found that transformational leadership shapes an organization's culture and practices, which also leads to high-performing teams (Forsgren et al., 2017). In particular, they identified the following practices as most significant (also see the classes on the left-hand side in Fig. 2).

- `Personal Recognition`: Consistently valuing and acknowledging achievements of outcomes or goals can be seen as a vital part in increasing the commitment of employees (Forsgren et al., 2017; Rafferty and Griffin, 2004).
- `Supportive Leadership`: Taking care of employees as a leader and taking into account their personal needs and feelings promotes a supportive work environment (Forsgren et al., 2017; Rafferty and Griffin, 2004).
- `Intellectual Stimulation`: An important factor of leadership is the ability of challenging employees to think about problems in new ways and generate an improved quality of solutions (Forsgren et al., 2017; Rafferty and Griffin, 2004).

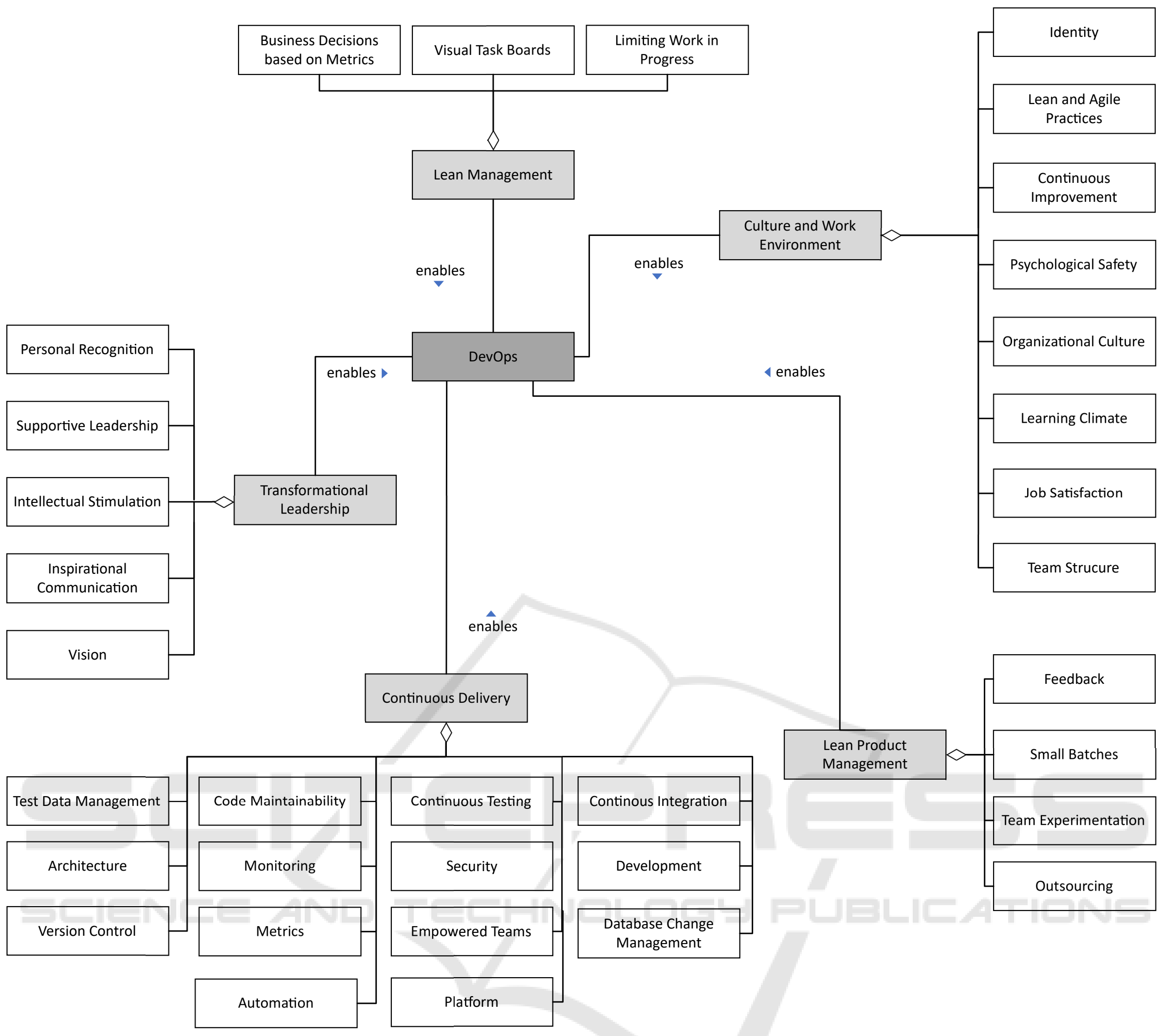

Figure 2: Concept map structuring key factors (goals) and sub-factors (sub-goals) enabling DevOps; oriented to the findings of the *State of DevOps Reports* by *DORA* 2014–2019 (Forsgren et al., 2019). For further details on the applied factors, see Section 3.2.

- `Inspirational Communication`: Being inspirational in the way the leaders of an organization communicate, builds motivation, confidence and stimulates the enthusiasm of employees (Forsgren et al., 2017; Rafferty and Griffin, 2004).
- `Vision`: Providing a vision to employees, which is ”The expression of an idealized picture of the future based around organizational values”, is an essential aspect of leadership. It ensures an understanding of both the organizational and team direction and accordingly where they should be in five years (Forsgren et al., 2017; Rafferty and Griffin, 2004).

Culture and Work Environment. DevOps strongly focuses on cultural practices that predict high organizational performance. E.g. in the 2014 *State of DevOps Report*, the researchers found that organizations with a good information flow, a high level of cooperation and trust, building bridging between teams, and a conscious inquiry culture, have a significant impact on the business performance of an organization (Forsgren et al., 2014). They identified the following practices as most significant. We extended them by further concepts identified in research literature (also see the classes on the right-hand side in Fig. 2).

- `Identity`: There is evidence, that when employees identify themselves with the organization they work for in terms of values, goals and apprecia-

tion, they tend to deliver higher levels of performance and productivity (Brown et al., 2016).
- `Lean and Agile Practices`: Adopting agile practices while choosing the practice combination and adoption strategies that would fit an organization´s priorities best, fosters knowledge and learning, employee satisfaction, social skill development as well as feedback and confidence (Solinski and Petersen, 2016).
- `Continuous Improvement`: The interaction of employees being able to reserve time for the improvement of their daily work, transparency with failures as well as feedback loops are key factors for a continuous improvement mindset (Kim et al., 2016).
- `Psychological Safety`: Whenever employees experience a culture of psychological safety, it is predictive to an increased software-delivery performance, organizational performance as well as overall productivity (Forsgren et al., 2019).
- `Organizational Culture`: *Dr. Westrum's* and *DORA* research findings about organizational culture show that also a high-trust generative culture predicts a better performance of software-delivery processes and overall productivity by influencing the way information flows through an organization (Forsgren, 2015).
- `Learning Climate`: Perceiving learning as an investment and key to improvement positively affects organizational culture and contributes to software-delivery performance improvements (Forsgren et al., 2018).
- `Job Satisfaction`: Employees with support by their team leaders and with adequate resources to get their work done provide better results and ultimately higher IT performance. Experiencing work as challenging and meaningful, while being empowered to exercise skills and judgment, positively adds to jobs satisfaction (Forsgren et al., 2014).
- `Team Structure`: Finding a team structure that fits best to an organization depends on the product set of an organization, the effectiveness of technical leadership, the willingness to change its IT-operations department and the skills to take the lead on operational concerns. An appropriate team structure should allow high cooperation and a shared responsibility between all functional areas (Skelton and Pais, 2019).

Lean Product Management. In the 2016 *State of DevOps Report*, the researchers found that organizations that adopt a lean approach to product design and delivery positively impact both IT performance and culture (Brown et al., 2016). They identified the following practices as most significant. We extended them by further concepts identified in research literature (also see the classes on the right-hand side in Fig. 2).
- `Feedback`: Including the feedback of customers into the design of products is of significant importance in order to meet the user expectations. Recognizing user feedback as a need and driver of change is essential for the software to evolve. A feedback cycle (i.e. collect, analyze, decide, and act) supports this process and helps avoiding that delivered functionalities are not needed (Brown et al., 2016).
- `Small Batches`: Teams which split products and features into small batches are able to release frequently, which allows them to gather customer feedback regularly (Brown et al., 2016).
- `Team Experimentation`: Empowering team members to interact with real users to learn about their needs, challenges and design solutions supports teams to add value by incorporating the learned inputs into the design of the product. Teams are able to work and make changes without having to ask for permission (Forsgren et al., 2018).
- `Outsourcing`: Whenever an organization decides to outsource a part of their application delivery to another provider, vendors need to be willing to partner, provide feedback and standardize application delivery and tooling between each other. Otherwise a DevOps-style model of collaboration is not realizable (Sharma, 2017).

Continuous Delivery. The *DORA* institute found that continuous delivery practices establish the basis for delivering value faster, thus ensuring shorter cycle times with quicker feedback loops, which improves the quality and overall organizational performance (Forsgren, 2015). They identified the following practices as most significant. We extended them by further concepts identified in research literature (also see the classes on the right-hand side in Fig. 2).
- `Test Data Management`: Managing test data is essential for running automated tests, resulting in lower change failure rates and fewer deployment issues (Brown et al., 2016).
- `Architecture`: Reducing the dependencies between teams, systems and testing/deploying activities allows teams to increase the deployment frequency and improve software-delivery performance (Forsgren et al., 2019).
- `Version Control`: Storing all kinds of code, configuration and scripts in a version control sys-

tem is an enabler for automation and continuous integration and predicts continuous delivery (Forsgren et al., 2014).

- `Code Maintainability`: Maintaining the code effectively is a vital practice to find, change, add, or reuse code from other teams and improves productivity by decreasing technical debt (Brown et al., 2016).
- `Monitoring`: Monitoring systems based on predefined metrics support teams to understand the state of their systems, guide business decisions and positively contributes to continuous delivery (Forsgren, 2015).
- `Metrics`: Additionally to the four most powerful metrics (i.e. deployment frequency, lead time for changes, time to restore service, and change-failure rate) there are many other possibly relevant monitoring metrics to focus on. Metrics should provide the possibility to predict an organization's ability to achieve its goals (Forsgren et al., 2019; Farshchi et al., 2018).
- `Automation`: Automated testing reduces the need to spend a lot of time finding and fixing bugs, wasting time and money on manual testing or releasing poor quality software (Humble and Farley, 2011). In particular, it reduces the risk of introducing errors into a previously correctly working software. Deployment automation reduces potential issues with deployments and provides fast feedback by allowing teams to comprehensively test as soon as possible after changes (Brown et al., 2016).
- `Continuous Testing`: Performing both automated and manual tests throughout the software-delivery lifecycle contributes to minimizing the likelihood of failure (Forsgren et al., 2018).
- `Security`: By better integrating security concerns into daily work and making it everyone´s responsibility, teams can achieve higher levels of software quality and build more secure systems (Humble and Farley, 2011; Forsgren et al., 2014).
- `Empowered Teams`: Teams that are empowered to choose their own tools and technologies tend to achieve higher software-delivery performance and have an increased job satisfaction (Forsgren et al., 2017).
- `Platform`: Teams need to be able to automatically provision new computing resources as needed, in order to focus on their goal of developing, testing, and delivering applications and services (Sharma, 2017; Forsgren et al., 2018).
- `Continuous Integration`: Frequently integrating all code in small logical units to a common repository at least once a day (verified by an automated build and test of each commit) enables rapid feedback loops and ensures that developers work in small batches, which contributes to the development of high-quality software (Fowler and Foemmel, 2006; Forsgren et al., 2014).
- `Development`: In order to achieve continuous integration and to eliminate complex merging events, small batches of code need to be integrated into trunk frequently (at least once a day) (Forsgren et al., 2017).
- `Database Change Management`: Integrating database changes into the deployment process reduces the risk and delay when performing a deployment (Forsgren et al., 2018).

4 MATURITY ASSESSMENT

In order to assess the maturity level based on the goals and sub-goals represented in the concept model in Section 3, we developed a survey-based technique. From a user perspective, first a questionnaire is answered, then the maturity report illustrating the individual and automatically measured levels of all applied capabilities can be checked. In the following, the questionnaire (Section 4.1), the maturity report (Section 4.2) as well as a first evaluation in terms of an expert survey 4.3 are presented.

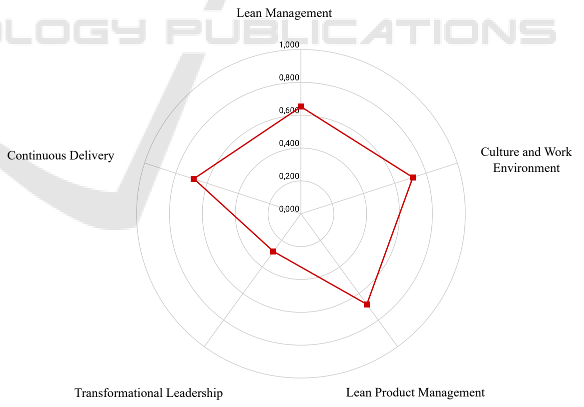

Figure 3: High-level DevOps maturity report in terms of a radar chart representing the calculated maturity levels of the 5 goals (key factors) enabling DevOps (see Section 4.2).

4.1 Questionnaire

The questionnaire comprises in total 98 questions, which cover and relate to the DevOps capabilities identified in Section 3.2 structured into goals and sub-goals in the class diagram in Fig. 2. The value of each sub-goal is assessed via several questions to be an-

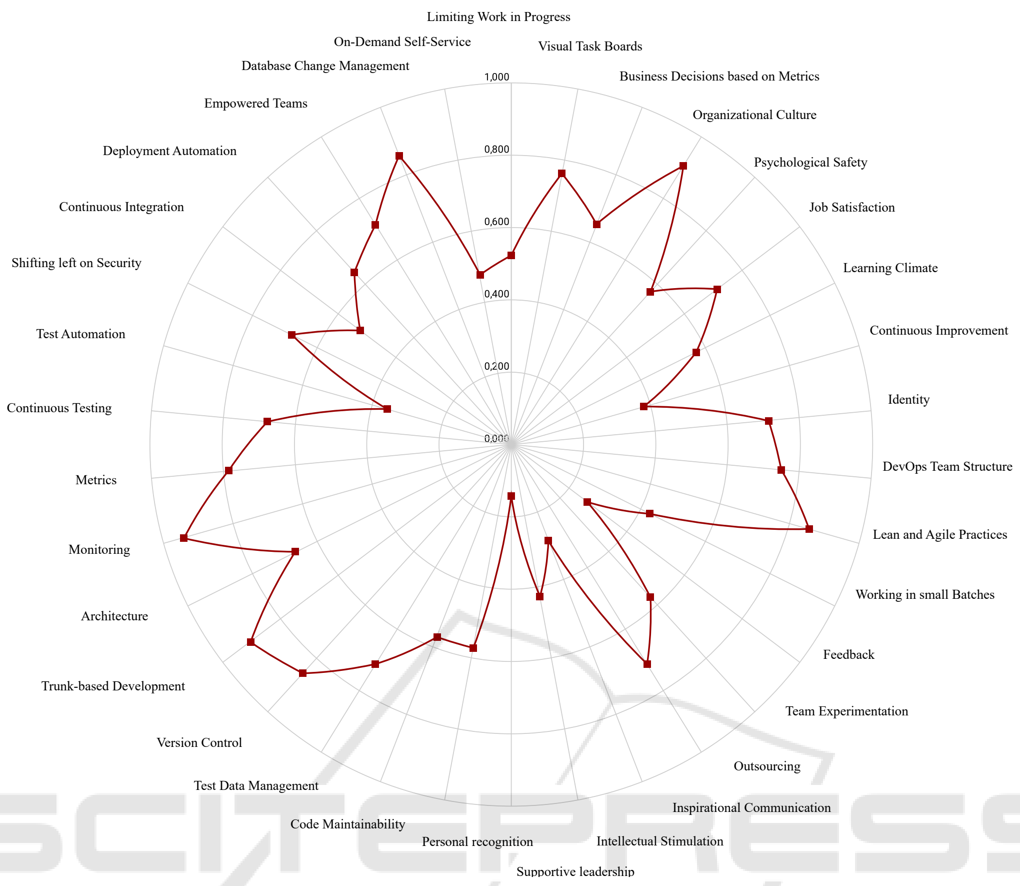

Figure 4: Detailed DevOps maturity report represented as a radar chart including the calculated values of 33 sub-goals (sub-factors) enabling DevOps (see Section 4.2).

swered by system stakeholders. The questions are either polar (a.k.a. Yes/No) questions or 1 to 7 linear-scale assessment questions (Likert, 1932). For calculating the resulting sub-goal-specific maturity levels, the questions are weighted proportionally (as well as the sub-goals for determining the values of the aggregating goals). The list of applied questions is presented in detail in Tables 1 and 2 in the Appendix.[2]

4.2 Maturity Report

The calculated individual results derived from the answers are then illustrated via different kinds of radar charts. Radar charts (a.k.a. spider charts) are two-dimensional charts structured by multiple quantitative variables for displaying multivariate data. For instance, they are popular means to visualize levels of competencies or quality metrics (Basu, 2004).

We apply the radar chart to illustrate the maturity levels of all DevOps goals or sub-goals identified in Section 3.2. Fig. 3 depicts an exemplary high-level maturity report representing the level of maturity for the 5 key goals. Moreover, Fig. 4 depicts an exemplary radar chart representing the levels of maturity of all applied 33 sub-goals enabling DevOps. Reviewing these reports provides a differentiated overview of the currently estimated DevOps maturity level. For example, it allows to identify fields or aspects that need further improvement or such that have already reached a relatively high level. Moreover, a comparison of different reports facilitates advanced analyses of the maturity level in an organization (see Section 5.2).

4.3 Survey

In order to evaluate and improve the model and assessment technique, we conducted a survey with 10 Software Engineering and DevOps experts and young professionals to test the practical utilization of the DevOps assessment. The respondents maintain different roles in the field of DevOps and belong to six different companies, each with a professional experience rang-

[2]The questionnaire has been created with *Google Forms* and is available at http://refactoringgames.org/devops/.

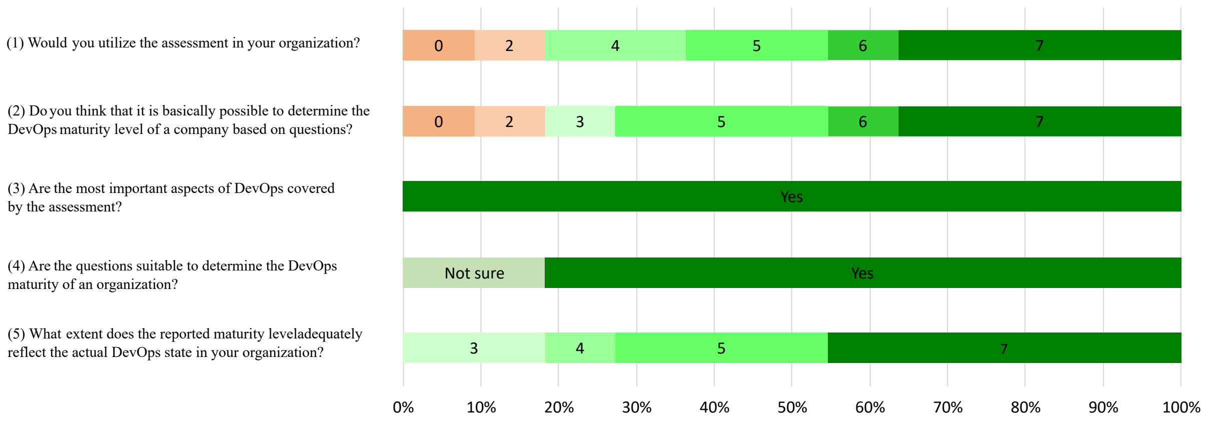

Figure 5: Results of the expert survey. For further details, see Section 4.3.

ing from 2 to more than 20 years.
The survey participants first answered the questionnaire (see Tables 1 and 2) and checked the resulting radar charts reporting the individually calculated maturity levels. The radar charts in Figs. 3 and 4 illustrate the average values of this survey. Subsequently, in order to evaluate the model and measurement technique, a second small survey has been conducted. A set of 8 questions has been defined to evaluate the questionnaire and the type of report (graphical radar chart). In this anonymous semi-structured survey also including open questions, participants have had the opportunity to provide a more in-depth feedback.

An overview of the answers to the closed-ended questions 1–5 of this expert survey is illustrated in Fig. 5 (questions 6–8 were open ended). Overall, the assessment was evaluated positively on its suitability to determine the DevOps maturity of an organization (see answers to questions 1 and 2 in Fig. 5). In particular, one survey respondent indicated that it should be differentiated whether the participant is a stakeholder of a software development organization or an IT service provider. All respondents noted that the DevOps practices mentioned in the survey were entirely representative of DevOps (see question 3 in Fig. 5). Moreover, it was found that a majority of respondents believes that it is basically possible to determine the DevOps maturity level of a company based on questions and furthermore would utilize the assessment in their organization (see question 4 in Fig. 5). Lastly, the majority of respondents agreed that the reported maturity level adequately reflect the actual DevOps state in their organization (see answers to question 5 in Fig. 5). Based on the feedback from the survey respondents, some details in the model and questionnaire have been slightly adjusted or added.

5 DISCUSSION

The presented assessment technique relies on the perspectives and estimations of the involved system stakeholders. With this in mind, we first reflect on the limitations of our approach (Section 5.1). Then we discuss further application opportunities in terms of potential use cases (Section 5.2).

5.1 Limitations

Assessing the DevOps maturity in organizations is challenging for several reasons. First, DevOps is a dynamic and not clearly defined principle which comprises concrete techniques, but also several cultural and organizational aspects that are more difficult to capture. It is possible that non-mentioned capabilities can be found in literature that are missing in the assessment. Second, conducting the DevOps assessment provides insights into the current state of DevOps implementation based on subjective perceptions of the participants via linear scale and polar questions. Technical aspects, such as the level of automation, could by measured by integrating evidence-based techniques, which could further increase the accuracy of the model. Anyway, assessing the cultural and (most of) organizational factors highly relies on the perspectives of the involved stakeholders. Thus, objectively measuring the maturity of DevOps remains a challenge. Third, the interpretation and implementation of DevOps in an organization is highly individual, as mentioned above. Therefore, the importance of goals and thus the weighting of questions can differ from organization to organization. Finally, as of now, we evaluated the assessment technique with 10 experts via one iteration. This first evaluation indicated that further iterations are needed in order to increase the practicability of the assessment.

5.2 Exemplary Use Cases

In addition to the application of the assessment technique to obtain a general evaluation of the DevOps maturity level, use cases for advanced analyses can be distinguished. These are essentially based on a comparison of several evaluations or reports and can provide answers to various questions. For each use case, the context and the possible implications for the organization are briefly described. In particular, we distinguish between *stakeholder-based analysis*, *cross-company analysis* and *impact monitoring*.

1. **Stakeholder-based Analysis:** First, consider conducting the survey with stakeholders related to a single company. The assessment will provide an overview of the different perspectives and evidence whether individuals in certain stakeholder roles have common or different understanding of the current state of DevOps. Besides the identification of deficit fields, the comparison might show significant inconsistencies regarding the state of certain aspects, which could be addressed by discussions or trainings.
2. **Cross-company Analysis:** Second, the survey can be utilized as benchmark within an industry. Comparing the DevOps maturity levels of multiple companies (or departments) can provide valuable insights into where the competition stands. This way, areas that might have been identified as weak spots can turn out to be above-average in relation to other companies.
3. **Impact Monitoring:** Third, orthogonally to the two use cases described above, the assessment technique could by applied before and after introducing new or modifying existing practices. By comparing the results, the effects of the performed measures can be analyzed, i.e. the perceived impact can be related to the invested resources (ROI analysis).

6 CONCLUSION

In this paper, we have presented a survey-based technique towards assessing DevOps maturity in organizations. Following a GQM-based process, 33 goals and sub-goals enabling DevOps have been structured, from which a set of 98 questions has then been derived. The DevOps maturity is represented via radar charts reporting the individual maturity levels for all identified (sub-) goals. A small evaluation in terms of a survey with 10 experts indicates that the developed assessment technique can provide support in analyzing the state of DevOps in organizations and in identifying focus areas that require more attention. Even though the first iteration received a predominantly positive echo from survey participants, the assessment offers opportunities for further improvements. As stated in Section 5.2, the assessment provides several practical use cases and provides a comprehensive view on DevOps practices. This information can also be useful to explain certain investments to management, especially when cultural aspects are involved.

In a next step, we plan to use our model and assessment technique on a larger scale to assess the maturity level of DevOps in organizations. For this purpose, we plan to apply the assessment technique within organizations that intend to enhance DevOps maturity and to distribute the survey to a larger number of participants. In addition, we prepare technical means to analyze the results according to the use cases described in Section 5.2, e.g. in terms of a stakeholder-based analysis and a cross-company analysis. Finally, we plan to provide tailor-made recommendations as feedback to enable the respondents to address the identified deficit areas accordingly and to initiate appropriate measures to increase the DevOps maturity level.

ACKNOWLEDGEMENTS

This work was partly supported by the Department for Economic Affairs, Labour and Statistics (MA23) of the City of Vienna (Austria) through the project ”New Work, New Business” at the UAS BFI Vienna.

REFERENCES

Bass, L., Weber, I., and Zhu, L. (2015). *DevOps: A software architect's perspective*. Addison-Wesley Professional.

Basu, R. (2004). *Implementing quality: a practical guide to tools and techniques: enabling the power of operational excellence*. Cengage Learning EMEA.

Becker, J., Niehaves, B., Poeppelbuss, J., and Simons, A. (2010). Maturity models in IS research. In *Proc. of ECIS 2010*.

Brown, A., Forsgren, N., Humble, J., Kersten, N., and Kim, G. (2016). State of DevOps report 2016.

Bucena, I. and Kirikova, M. (2017). Simplifying the DevOps adoption process. In *BIR Workshops*.

Caldiera, V. R. B. G. and Rombach, H. D. (1994). The goal question metric approach. *Encyclopedia of software engineering*, pages 528–532.

Farcic, V. (2019). *DevOps paradox: The truth about DevOps by the people on the front line*. Packt Publishing, Birmingham, UK.

Farshchi, M., Schneider, J., Weber, I., and Grundy, J. (2018). Metric selection and anomaly detection for cloud operations using log and metric correlation analysis. *Journal of Systems and Software*, pages 531–549.

Fenton, N. and Bieman, J. (2014). *Software metrics: a rigorous and practical approach*. CRC press.

Forsgren, N. (2015). State of DevOps report 2015.

Forsgren, N., Gene, K., Nigel, K., and Jez, H. (2014). State of DevOps report 2014.

Forsgren, N., Gene, K., Nigel, K., Jez, H., and Brown, A. (2017). State of DevOps report 2017.

Forsgren, N., Humble, J., and Kim, G. (2018). Accelerate: State of DevOps report 2018.

Forsgren, N., Smith, D., Humble, J., and Frazelle, J. (2019). Accelerate: State of DevOps report 2019.

Fowler, M. and Foemmel, M. (2006). Continuous integration.

Gruver, G. (2016). *Starting and scaling DevOps in the enterprise*. Gary Gruver, [United States].

Gruver, G. (2019). *Engineering the Digital Transformation*. Gruver, 1 edition.

Gruver, G., Mouser, T., and Kim, G. (2015). *Leading the transformation: Applying Agile and DevOps principles at Scale*. IT Revolution, Portland, OR.

Harrison, D., Lively, K., and Wang, A. (2019). *Achieving DevOps: A novel about delivering the best of Agile, DevOps, and microservices*. Apress, New York, NY.

Hevner, A. R., March, S. T., Park, J., and Ram, S. (2004). Design science in information systems research. *MIS quarterly*, pages 75–105.

Humble, J. and Farley, D. (2011). *Continuous delivery: Reliable software releases through build, test, and deployment automation*. Addison-Wesley, Upper Saddle River, NJ.

Kim, G., Humble, J., Debois, P., and Willis, J. (2016). *The DevOps Handbook: How to Create World-Class Agility, Reliability, and Security in Technology Organizations*. IT Revolution.

König, L. and Steffens, A. (2018). Towards a quality model for DevOps. *Continuous Software Engineering & Full-scale Software Engineering*, page 37.

Lasrado, L., Vatrapu, R., and Andersen, K. (2015). Maturity models development in IS research: A literature review. volume 6. Scandinavian Chapter of the Association for Information Systems (AIS) - Scandinavian IRIS. 38th Information Systems Research Seminar in Scandinavia, IRIS38 ; Conference date: 09-08-2015 Through 12-08-2015.

Likert, R. (1932). A technique for the measurement of attitudes. *Archives of psychology*.

Linda and ling Lai, S. (2012). Managing user expectations in information systems development. *International Journal of Economics and Management Engineering*, 6(12):3710 – 3714.

Mettler, T. (2011). Maturity assessment models: a design science research approach. *International Journal of Society Systems Science (IJSSS)*, 3(1/2):81–98.

Morales-Ramirez, I., Perini, A., and Guizzardi, R. (2015). An ontology of online user feedback in software engineering. *Applied Ontology*, 10:297–330.

Object Management Group (2017). Unified Modeling Language (UML), Superstructure, Version 2.5.1. https://www.omg.org/spec/UML/2.5.1 [July 31, 2019].

Peffers, K., Tuunanen, T., Rothenberger, M. A., and Chatterjee, S. (2007). A design science research methodology for information systems research. *Journal of management information systems*, 24(3):45–77.

Poeppelbuss, J., Niehaves, B., Simons, A., and Becker, J. (2011). Maturity models in information systems research: literature search and analysis. *Communications of the Association for Information Systems*, 29(1):27.

Radstaak, J. (2019). Developing a DevOps maturity model: A validated model to evaluate the maturity of DevOps in organizations. Master's thesis, University of Twente.

Rafferty, A. E. and Griffin, M. A. (2004). Dimensions of transformational leadership: Conceptual and empirical extensions. *The Leadership Quarterly*, 15(3):329–354.

Sharma, S. (2017). *The DevOps adoption playbook: A guide to adopting DevOps in a multi-speed IT enterprise*. Wiley, Indianapolis IN.

Skelton, M. and Pais, M. (2019). *Team topologies: Organizing business and technology teams for fast flow*. IT Revolution, Portland, Oregon.

Smeds, J., Nybom, K., and Porres, I. (2015). Devops: a definition and perceived adoption impediments. In *International Conference on Agile Software Development*, pages 166–177. Springer.

Solinski, A. and Petersen, K. (2016). Prioritizing agile benefits and limitations in relation to practice usage. *Software Quality Journal*, 24(2):447–482.

van Hillegersberg, J. (2019). The need for a maturity model for maturity modeling. In *The Art of Structuring*, pages 145–151. Springer International Publishing, Cham.

van Ommeren, E., van Doorn, M., Dial, J., and van Herpen, D. (2016). Mastering digital disruption with DevOps: Design to disrupt.

von Wangenheim, C. G., Hauck, J. C. R., Salviano, C. F., and von Wangenheim, A. (2010). Systematic literature review of software process capability/maturity models. In *Proceedings of International Conference on Software Process Improvement and Capabity Determination (SPICE), Pisa, Italy*.

Wendler, R. (2012). The maturity of maturity model research: A systematic mapping study. *Information and software technology*, 54(12):1317–1339.

Westrum, R. (2004). A typology of organizational cultures. *Quality and Safety in Health Care*, 13:22–27.

Table 1: Applied goals, sub-goals and corresponding questions for assessing DevOps maturity (part 1).

DevOps Assessment			
Goal	Sub-Goal	Question	Source
Lean Management	Limit Work in Progress	To what extent are teams able to limit work in progress (WIP) and use these limits to drive process improvement and increase throughput?	(Forsgren, 2015)
	Limit Work in Progress	To what extent are teams aware of the mean lead time and variability the entire value stream (from idea to customer)?	(Forsgren, 2015)
	Limit Work in Progress	How often does your team have to leave unfinished work to start working on another requirement?	(Forsgren, 2015)
	Visual Boards	To what extent do teams create and maintain visual displays (f.e. boards) that are showing key quality and productivity metrics and the current status of work (including defects)?	(Forsgren, 2015)
	Visual Boards	To what extent are the visual displays giving you the information you need?	(Forsgren, 2015)
	Visual Boards	To what extent is the information on visual displays up to date and are people acting on this information?	(Forsgren, 2015)
	Business Decisions	To what extent do teams use data from application performance and infrastructure monitoring tools to make business decisions on a daily basis?	(Forsgren, 2015)
Culture and Work Environment	Westrum Org Perf.	On my team, information is actively sought.	(Westrum, 2004)
	Westrum Org Perf.	On my team, failures are learning opportunities, and messengers of them are not punished.	(Westrum, 2004)
	Westrum Org Perf.	On my team, responsibilities are shared.	(Westrum, 2004)
	Westrum Org Perf.	On my team, cross-functional collaboration is encouraged and rewarded.	(Westrum, 2004)
	Westrum Org Perf.	On my team, failure causes enquiry.	(Westrum, 2004)
	Westrum Org Perf.	On my team, new ideas are welcomed.	(Westrum, 2004)
	Psychological Safety	To what extent do team members feel safe to take risks and be vulnerable in front of each other?	(Forsgren et al., 2019)
	Job Satisfaction	To what extent do you see your work as challenging and meaningful?	(Forsgren et al., 2014)
	Job Satisfaction	To what extent do you feel empowered to exercise your skills and judgment at work?	(Forsgren et al., 2014)
	Learning Culture	To what extent does your organization view learning as the key to improvement?	(Forsgren et al., 2014)
	Learning Culture	To what extent does your organization see learning as an investment rather than an expense?	(Forsgren et al., 2014)
	Continuous Improvement	To what extent are team leaders and managers creating conditions for employees to improve their daily work and emphasize learning and problem solving?	(Kim et al., 2016)
	Continuous Improvement	To what extent are employees able to reserve time for the improvement of their daily work?	(Kim et al., 2016)
	Continuous Improvement	To what extent are failures reported and made transparent throughout the organization in order to gain experience to prevent incidents from happening again?	(Kim et al., 2016)
	Job Identity	To what extent do you identify yourself with the organization you work for (values, goals, appreciation)?	(Brown et al., 2016)
	Team Structure	To what extent are aspects like the product set of an organization, anti-types (bad practices) and established DevOps team structures considered when forming the team structure of the organization?	(Skelton and Pais, 2019)
	Team Structure	To what extent does the current organizational structure allow high cooperation between all functional areas, especially those that are included in developing the end-product?	(Skelton and Pais, 2019)
	Team Structure	To what extent do teams have a shared responsibility between each functional area of the software delivery process with regards to building, deploying, and maintaining a product?	(Skelton and Pais, 2019)
	Team Structure	To what extent is software development managed like projects, meaning that developers get assigned to the next project immediately after the project goal is met?	(Skelton and Pais, 2019)
	Agile	To what extent are agile values and principles established within the organization?	(Gruver, 2019)
Lean Product Management	Small Batches	To what extent do teams slice up products and features into small batches that can be completed in less than a week and released frequently, including the use of minimum viable products (MVPs)?	(Forsgren et al., 2018)
	Feedback	To what extent does your organization actively and regularly seek customer feedback and incorporate this feedback into the design of their products?	(Forsgren et al., 2018)
	Team Experimentation	To what extent do development teams have the authority to create and change specifications as part of the development process without requiring approval?	(Forsgren et al., 2018)
	Software Evolution	To what extent is the role of user feedback recognized as a need of change?	(Morales-Ramirez et al., 2015)
	Software Evolution	To what extent is user feedback relevant for software evolution (f.e. new designs evolving from old ones)?	(Morales-Ramirez et al., 2015)
	Feedback Cycle	To what extent is a feedback cycle (Collect, Analyse, Decide, Act) implemented as a process?	(Morales-Ramirez et al., 2015)
	Expectation-Perception Gap	To what extent does the user´s perception of the delivered system meet the the user´s expectations?	(Linda and ling Lai, 2012)
	Expectation-Perception Gap	How often a customer/business does not need/use the delivered functionalities or use the ordered features very rarely?	(Bucena and Kirikova, 2017)
	Outsourcing	Does your organization outsource whole functions such as application development, IT operations work, or testing and QA?	(Sharma, 2017)
	Outsourcing	To what extent are vendors not willing to partner because the contracts in place do not provide for a DevOps-style model of collaboration?	(Sharma, 2017)
	Outsourcing	Does your team build applications in-house and deliver them to a production environment managed by an external vendor?	(Sharma, 2017)
	Outsourcing	To what extent does your team receive appropriate feedback from the vendor to improve continuously?	(Sharma, 2017)
	Outsourcing	To what extent is your team partnering closely with the organization on standardizing application delivery and tooling?	(Sharma, 2017)
Transformational Leadership	Vision	To what extent do leaders in your organization have a clear concept of where the organization is going and where it should be in five years?	(Forsgren et al., 2017)
	Inspirational communication	To what extent do leaders in your organization communicate in a way that inspires and motivates, even in an uncertain or changing environment?	(Forsgren et al., 2017)
	Intellectual stimulation	To what extent do leaders in your organization challenge their teams to think about problems in new ways (ask new questions, status quo, basic assumptions about the work)?	(Forsgren et al., 2017)
	Supportive leadership	To what extent do leaders in your organization demonstrate care and consideration of followers' personal needs and feelings?	(Forsgren et al., 2017)
	Personal recognition	To what extent do leaders in your organization praise and acknowledge achievement of goals and improvements in work quality; personally compliment others when they do outstanding work?	(Forsgren et al., 2017)

Table 2: Applied goals, sub-goals and corresponding questions for assessing DevOps maturity (part 2).

DevOps Assessment			
Goal	Sub-Goal	Question	Source
Continuous Delivery	Code Maintainability	To what extent do teams manage code maintainability (systems and tools that make it easy for developers to change code maintained by other teams, find examples in the codebase, reuse other people's code, as well as add, upgrade, and migrate to new versions of dependencies without breaking their code)?	(Forsgren et al., 2017)
	Test Data Management	To what extent is adequate test data available for their work?	(Brown et al., 2016)
	Test Data Management	To what extent can test data be acquired on demand for test suites?	(Brown et al., 2016)
	Version Control / Application code	Do you use version control for application code?	(Forsgren, 2015)
	Version Control / Application code	How easily and quickly can a team recover application code from the version control system?	(Forsgren, 2015)
	Version Control / System configurations	Do you use version control for system configurations?	(Forsgren, 2015)
	Version Control / System configurations	How easily and quickly can teams reconfigure systems from version control?	(Forsgren, 2015)
	Version Control	To what extent do you use as few repositories as possible for all requirement specifications and related documentation?	(Bucena and Kirikova, 2017)
	Trunk-based development	To what extent are teams aware of how many active branches they have on their application repositories' version control system and aims to reduce the branches?	(Forsgren et al., 2017)
	Trunk-based development	To what extent are teams aware of how many code freezes they have and how long they last, and aims to reduce them?	(Forsgren et al., 2017)
	Trunk-based development	To what extent are teams aware of the number of branches and forks that are merged per day to master and aims to increase the frequency?	(Forsgren et al., 2017)
	Loosely coupled architecture	To what extent are teams able to independently test, deploy, and change their systems on demand without dependingon other teams for additional support, services, resources, or approvals?	(Forsgren et al., 2019)
	Architecture	To what extent are the existing architectures facilitating the practices required for improving software delivery performance (increased deployment frequency with reduced lead time for changes, time to restore service, and change failure rate)?	(Forsgren et al., 2017)
	Architecture	To what extent are major architectural archetypes considered when it comes to selecting the architecture for a new system?	(Forsgren et al., 2017)
	Monitoring	To what extent is data from application performance monitoring tools used to support business decisions?	(Forsgren, 2015)
	Monitoring	To what extent is data from infrastructure monitoring tools used to support business decisions?	(Forsgren, 2015)
	Monitoring / Proactive Failure Notifications	To what extent are failure alerts from logging and monitoring systems captured and used?	(Forsgren et al., 2018)
	Monitoring / Proactive Failure Notifications	To what extent is system health proactively monitored using threshold warnings?	(Forsgren et al., 2018)
	Monitoring / Proactive Failure Notifications	To what extent is system health proactively monitored using rate of change warnings?	(Forsgren et al., 2018)
	Metrics	How often does your organization deploy code to production or release it to end users?	(Forsgren et al., 2019)
	Metrics	What is your lead time for changes (i.e., how long does it take to go from code committed to code successfully running in production)?	(Forsgren et al., 2019)
	Metrics	How long does it generally take to restore service when a service incident or a defect that impacts users occurs (e.g., unplanned outage or service impairment)?	(Forsgren et al., 2019)
	Metrics	What percentage of changes to production or released to users result in degraded service (e.g., lead to service impairment or service outage) and subsequently require remediation (e.g., require a hotfix, rollback, fix forward, patch)?	(Forsgren et al., 2019)
	Continuous Testing	Do the services developed by your team have a production-like environment available for QA and/or operations?	(Forsgren et al., 2018)
	Continuous Testing	How often does it happen that critical errors are only discovered in the production environment?	(Forsgren et al., 2018)
	Continuous Testing	Is there a standardized tool and process to track issues/defects?	(Forsgren et al., 2018)
	Continuous Testing	To what extent are teams continuously reviewing and improving test suites to better find defects and keep complexity and cost under control?	(Forsgren et al., 2018)
	Continuous Testing	To what extent are testers allowed to work alongside developers throughout the software development and delivery process?	(Forsgren et al., 2018)
	Continuous Testing	To what extent are teams performing manual test activities such as exploratory testing, usability testing, and acceptance testing throughout the delivery process?	(Forsgren et al., 2018)
	Continuous Testing	To what extent do developers practice test-driven development by writing unit tests before writing production code for all changes to the codebase?	(Forsgren et al., 2018)
	Continuous Testing	Are teams able to get feedback from automated tests in less than ten minutes both on local workstations and from a CI server?	(Forsgren et al., 2018)
	Test Automation	To what extent does the time spent on fixing test failures changes over time?	(Forsgren et al., 2018)
	Test Automation	How often do automated test failures represent a real defect (in contrast to being poorly coded)?	(Forsgren et al., 2018)
	Test Automation	Do all test suites run in every pipeline trigger?	(Forsgren et al., 2018)
	Test Automation / Security	To what extent do automated tests cover security requirements?	(Brown et al., 2016)
	Shifting left on Security	To what extent do features undergo a security review early in the design process?	(Brown et al., 2016)
	Shifting left on Security	To what extent do security reviews slow down the development cycle?	(Brown et al., 2016)
	Shifting left on Security	To what extent is the security team involved in each step of the software delivery lifecycle (design, develop, test, and release)?	(Brown et al., 2016)
	Continuous Integration	To what extent do code commits result in a software build without manual intervention?	(Forsgren, 2015)
	Continuous Integration	To what extent do code commits result in a suite of automated tests being run without manual intervention?	(Forsgren, 2015)
	Continuous Integration	To what extent are automated builds and automated tests successfully executed every day?	(Forsgren, 2015)
	Continuous Integration	To what extent are builds available to testers?	(Forsgren, 2015)
	Continuous Integration	To what extent is feedback from acceptance and performance tests available to developers within a day?	(Forsgren, 2015)
	Continuous Integration	How long does it take between a build breaking and having it fixed, either with a check-in that fixes the problem, or by reverting the breaking change?	(Forsgren, 2015)
	Deployment Automation	To what extent are teams aware of the number of manual steps in the deployment process and aim to reduce those steps?	(Brown et al., 2016)
	Deployment Automation	To what extent is a delivery pipeline (or deployment pipeline) implemented that automates the orchestrating of deployments on all necessary components?	(Bucena and Kirikova, 2017)
	Empowered Teams	To what extent is a development team allowed to change requirements or specifications in response to what they discover, without authorization from some outside body?	(Forsgren et al., 2017)
	Database Change Management	To what extent are database changes integrated into the software delivery process (configuration management, communication with DBAs)?	(Forsgren et al., 2018)
	On-Demand Self-Service	To what extent are teams able to automatically provision computing resources as needed, without human interaction from the provider?	(Forsgren et al., 2019)
	On-Demand Self-Service	To what extent does the infrastructure platform automatically control, optimize, and report resource use based on the type of service such as storage, processing, bandwidth, and active user accounts?	(Forsgren et al., 2019)
	On-Demand Self-Service	How long does it take to get up an HelloWorld! application in an environment using the standard processes?	(Gruver, 2016)