Measuring Agile/DevOps team performance

Thoby Visser[1], Joris Hulstijn[2]

[1]Capgemini, Utrecht, Netherlands
[2]University of Luxembourg, Esch sur Alzette, Luxembourg

Abstract

The increasing adoption of Agile and DevOps has triggered the following question: "How can the performance of Agile/DevOps teams be continuously improved?" In this paper we analyze how combined Agile/DevOps team performance can be appropriately measured and quantified, and identify factors that have an influence on team performance. A literature review was conducted to identify metrics for measuring team performance, combining both Agile and DevOps. Semi-structured interviews were conducted with nine experts in Capgemini, who manage multiple Agile/DevOps teams. They were asked to propose and substantiate factors that are deemed important to team success. These factors are aggregated and validated, and structured in a conceptual model. The resulting performance metrics and factors are compiled into a DMAIC cycle adaptation model, which focuses on providing an actionable process to continuously improve team performance. The method is currently being tested in practice

Keywords

Metrics, Agile, DevOps, Performance

1. Introduction

Adoption of Agile methodologies has experienced dramatic growth in the last decade [1]. Agile is a family of widely adopted software development methods, promoted through the Agile Manifesto [2], which was developed to gain the ability to 'create and respond to change' [3]. This enables teams to streamline the development process and facilitate changing business requirements [4]. Also DevOps has seen significant growth in usage in recent years [5]. DevOps can be seen as a natural evolution of the Agile development methods [6], by integrating the operations practice, using automated development, deployment, and infrastructure monitoring [7]. DevOps aims to bring development (Dev) and operations (Ops) activities together in one team, to reduce the time between committing a change and the change being placed into production, while ensuring high quality and stability [8].

Teams are central to software development. There are clear differences in the performance of teams [9]. Some differences, can be attributed to the members of a team (experience, social skills), but not all. There is a team effect [10]. If one team develops a new way of working that is successful, other teams may benefit too. Which factors affect team performance? Can team performance be improved? What do we mean by team performance for combined Agile/DevOps teams? For example, do we focus on speed, or do we focus on reliability of the software? Can team performance be measured, reliably? These are the topics we study in this paper.

2nd International Workshop on Agile Methods for Information Systems Engineering (Agil-ISE'23)
thoby.visser@capgemini.com (T. Visser); joris.hulstijn@uni.lu (J. Hulstijn)

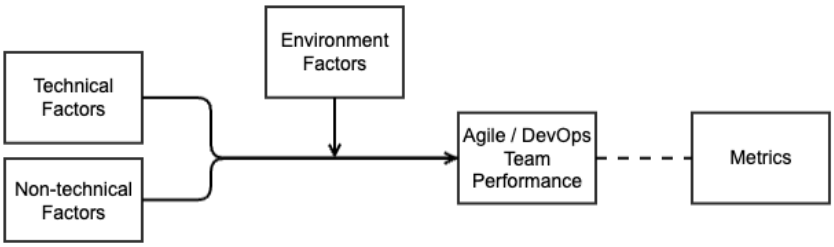

Figure 1: Research Model

RQ. How can the performance of Agile/DevOps teams be continuously improved?
1. What metrics represent the performance of the Agile/DevOps teams?
2. What factors impact the performance of the Agile/DevOps teams?

This paper only provides a brief summary of [11]. The research approach is design science [12]. The idea is to (1) improve a problem context, by (2) (re-)designing an artefact, that (3) satisfies some requirements, in order to (4) help stakeholders achieve some goals, see template [12, p 16].

(1) The problem context is as follows: at GapGemini, like at other IT providers, combined Agile/DevOps teams are deployed at clients. Teams are managed remotely. Because of the autonomy of teams in Agile methods, it is hard to monitor performance. Management is seeking methods to help teams improve their performance. (2) An artefact is developed: an improvement method. After deliberation, we chose to develop the so called DMAIC framework: define - measure - analyse - improve - control [13]. DMAIC is based on repeated cycles of reliable measurements, learning and improvement. This method underlies the Lean Six Sigma methodology. (3) The metrics that are part of the DMAIC method must meet a number of requirements. They must be relevant, measurable, reliable and comparable [14]. (4) The goal of the stakeholders is to continuously improve team performance.

The research proceeds in four steps. First, based on a literature review, the notion of team performance was studied, for both Agile and DevOps, identifying potential factors that influence team performance. Second, based on a review of the literature on metrics in general, and metrics of software development and deployment in particular, a set of potential metrics was proposed. Third, semi-structured interviews were conducted with nine experts in CapGemini, who manage multiple Agile/DevOps teams. These people can be seen as experts, both on the factors that contribute to team performance, and on measurements and monitoring. The interviewees were asked to validate the factors, and also the metrics. Fourth, the resulting set of factors and metrics, were combined in a DMAIC framework, to provide guidance to stakeholders.

The remainder of the paper is structured as follows. Section 2 details factors that have an influence on team performance. Section 3 discusses the set of possible metrics. Finally, Section 4 presents a brief overview of the DMAIC framework. The paper ends with discussion of future research and conclusions.

Factor	N	Definition
		Technical Factors
(Test) Automation	38	Automating tasks and processes
Expertise	26	Relevant technical skills and knowledge
(Collaborative) Tools	13	Applications used in the Agile/DevOps lifecycle and to collaborate
T-Shaped	10	Ability to apply knowledge across situations, and having specific expertise too.
Technical Debt	10	Complexity of a software system that makes development more difficult, often accumulated by choosing simple fixes over suitable solutions
Facilities	8	A suitable workspace with equipment
Business Orientation	8	To take business perspective into account
Technical Overview	5	Ability to oversee complex technical landscapes (systems and infrastructure).
		Non-technical Factors
Retention	41	The time the team members have been together and continue to stay in a team
Collaboration	23	Interpersonal cooperation of team members
Team Climate	20	Culture and environment among the members of a team
Common Goal	16	An objective worked towards by all team members
Seniority	13	Balance of employer expertise, defined by skill and length of service.
Team Diversity	15	Variety in character, expertise and roles.
Distributed Team	10	Teams that are geographically spread out.
Independence	10	Ability to solve solutions without assistance from outside the team.
Trust (Inside)	8	Belief of reliability, truth, or ability within the team and between its members.
Soft Skills	8	Ability to interact effectively with others.
Respect	5	Respect for the other team members.
Workload Focus	5	Team members are focused on a single project and not multiple projects.
Transparency	5	Honesty and openness between team members.
		Environment Factors
Business-IT Alignment	36	Effective communication and a shared understanding of business and IT.
Autonomy	20	Ability and freedom for teams to manage, govern and organise themselves.
Planning Stability	16	No change in planned business requirements during an iteration.
Technological Innovation	16	The implementation of new technologies.
Acceptance	15	Ability of the organisation to accept Agile/DevOps methodology.
Project Planning	15	Prioritisation, selection and control of the business requirements for projects.
Business Vision	10	Strategic goals, values and aspirations.
Readiness	10	Prepared to accept change in culture/methodology.
Trustworthiness	8	Perceived to be reliable, sincere and competent.
Facilitation	5	Be enabled to operate by other departments.
Knowledge Sharing	5	Sharing of knowledge and information.

Table 1
Factors characterizing high performing teams and team members from the expert interviews. The N score is calculated by summing the experts' ranks of importance per factor on a scale of $\langle 10, 5, 3, 2, 1 \rangle$.

2. Factors

A systematic literature review of the Agile way of working, DevOps, and team performance [11, Ch 2] produced a list of factors, which were discussed in semi-structured interviews. A total of 9 interviews were held. Interviewees were selected among employees of GapGemini because of their role (service coordinator, service delivery manager, project manager) and expertise. Each interviewee manages between 2 and 6 teams. The application domains of the clients are public sector (5 times), retail (1), utilities (1) and energy (1). Client size ranges from 500-2500 (4), 2500-5000 (2) to 5000-77500 (2) employees. The experts were asked to rank the relative importance of factors, on a scale of $\langle 10, 5, 3, 2, 1 \rangle$. The outcome is shown in Table 1.

Stability:	
Change failure rate:	Number of failed deployments in production in a given period.
Time to restore service:	Average time taken to restore a running application or service since the issue was detected.
Mean time to recovery (MTTR):	Total time taken to recover the applications or services or systems for all failures, divided by number of failures
Defect density:	Number of confirmed defects detected in software in a given period, divided by the size of the software or component.
Throughput:	
Deployment frequency:	Number of deployments/releases in production in a given period.
Change lead time:	The length of time between when a code change is committed/delivered to when it is deployed to production [15].
User stories planned versus delivered:	Completed number of story points divided by planned number of story points.
Mean time to resolve:	Average time taken to develop a fix and roll out in production.

Table 2
Selection of most cited and suitable performance metrics for stability and throughput

The most important technical factors are automation, the extent to which tasks and processes are automated in a CI/CD pipeline, and technical skills and knowledge. Consider the following quote: "Automation is important. The automation of processes and the [CI/CD] pipeline. Not only can it save a lot of useful time, but it also makes the team members think of continuous improvement." (interviewee I). Non-technical factors are seen as relatively more important. Consider factors like retention, collaboration, team climate, and team diversity. Regarding retention, one of the interviewees remarked: "The longer a team is together, the better they will be tuned in to each other. They will be able to better understand each other and know what the others have to offer. " (interviewee H). With regard to the environment in which the team must operate, the most important factors are business-IT alignment, and autonomy of teams, according to the interviewees: "There should be a shared understanding of both business and IT. Working with DevOps is not very traditional and undoubtedly requires the business to adapt. Both parties need to be on the same page for it to work." (interviewee C)

3. Metrics

Agile methods focus on change [10]. Improved communication in a team and with stakeholders and delivering software frequently, allow teams to quickly receive feedback and improve. This increases the speed of development. However, a focus on speed may come at a cost in terms of quality. Faults may lead to delays later. Operations people care more about stability of the entire system. The core of DevOps is that this tension between speed in development and stability in operations, is brought into the team [8]. Quality and reliability concerns are resolved at an earlier stage of development. Teams that can handle this tension, are seen as successful teams.

We conducted a systematic literature review about metrics in general, and metrics for Agile and DevOps team performance [Ch 2.3][11]. Metrics are obtained from the 'State of Agile' report [16], and from a review of DevOps performance metrics. The research produced a list of 34 potential performance metrics. Out of these, 21 metrics refer to *throughput*, so they are applicable to both Agile and DevOps, and 13 metrics focus on *stability* and are applicable to DevOps. A summary of suitable metrics for stability and throughput is listed in Table 2.

Given these metrics, it is possible to formulate an equation summarizing team performance, over a given period: $P(t) = (w_T \cdot T(t)) \cdot (w_S \cdot S(t))$, where $P(t)$ is performance for team t, $T(t)$ and $S(t)$ are throughput and stability metrics respectively, and w_T and w_T are the relative weights for throughput and stability in a given project, such that $w_T + w_S = 1.0$.

Metrics have to meet certain requirements: they must be *relevant* to the business and to Agile and DevOps team performance, they must be *measurable*, i.e. have standardized values that are consistent over time [14], *reliable*, i.e. verifiable and free from bias, and *comparable*, so they can be used to see if performance is getting better or worse under different conditions.

To make sure that metrics are measured consistently over time and their values have real meaning, the measurement must be well-integrated in existing software development and project management processes, and be taken directly from information systems.

For these reasons, the metric of *velocity* was not selected in Table 2. Velocity is how much technical scope a team can develop over a period of time. Velocity is usually expressed in story points per period, an estimate of the expected effort required to implement a piece of work. Story points are subjective by nature and therefore do not adhere to the criteria of comparability and reliability [17]. However, velocity is widely used in the Agile community (see below); so it may still be included to accommodate an established way of working. An alternative to estimate the effort needed for a piece of software is to use automated function points (AFPs) [18].

The adoption of such metrics was discussed with the experts. Although various interviewees stated that they were already tracking metrics (7 said so), most of them said their use was limited. The metrics that are collected are velocity (mentioned 7 times), change lead time (4), number of defects (1), and employee turnover (1). Several interviewees mentioned that their client has shown interest in establishing metrics to measure performance (5). Not all interviewees support independently measuring team performance: "Metrics do not exhibit trust in the teams, which are supposed to be autonomous and self-organising." (Interviewee *I*).

4. Framework

For this research, the DMAIC cycle is used as a framework for facilitating continuous performance improvement within the context of Agile/DevOps teams. DMAIC is also the methodology underlying the well known Lean six Sigma quality improvement method. The DMAIC cycle is chosen over other improvement methodologies – like plan-do-check-act, RADAR or DFSS – because of the data-driven orientation [19]. It contains a specific step for measurement. This fits the original goal to monitor team performance, and to evaluate which interventions to improve factors, actually work. The main steps are summarized and illustrated with an example.

1. *Define*: define the goal. Here, the goal is to improve Agile/DevOps team performance.
2. *Measure*: measure team performance, using the proposed metrics (Table 2).
   Example: the 'defects over time' metric has increased compared to earlier iterations.
3. *Analyse*: identify problem areas. Select which factors impacting performance can be adjusted and possibly improved (Table 1).
   Example (cont.): Analysis shows that the increase in 'defects over time' metric is caused by the complexity of the software. The complexity is accumulated by many simple fixes. In response, the 'technical debt' factor is identified for improvement.

4. *Improve*: start projects or interventions to adjust and manage the identified factors.
   Example: start a project to reduce complexity, by refactoring the software in crucial modules, replacing quick fixes with a principled solution.
5. *Control*: verify whether the projects lead to performance improvements, by using the metrics, and whether adjusted factors are controlled and sustained.
   Example: 'defects over time' and 'technical debt' are monitored. If technical debt remains a problem, additional modules can be refactored.

5. Conclusions

The aim of this research is to find a method to continuously improve team performance in Agile/DevOps teams. A possible method is the DMAIC framework: define, measure, analyze, improve and control. In such continuous improvement methods, team performance must be made measurable and quantifiable. Additionally, factors that may impact the team performance must be identified. Projects can then be started to try and adjust these factors.

Based on a literature study a list of metrics is generated, to measure team performance. The metrics are obtained from the 'State of Agile' report [16] and from a literature review on DevOps performance metrics. The metrics are verified against the criteria: relevance, measurability, reliability and comparability. A summary of suitable metrics is listed in Table 2.

Semi-structured interviews are conducted with 9 experts in Capgemini who each manage multiple Agile/DevOps teams. The experts are interviewed on topics like experiences, adoption, team performance and metrics. The experts are asked to propose and substantiate 'technical', 'non-technical' and 'environmental' factors that affect team performance, according to them. These factors are aggregated, ranked and linked to literature. The factors are shown in Table 1.

The resulting performance metrics and factors affecting team performance are combined in the DMAIC cycle, adjusted to the context of Agile/DevOps team performance. A full scale evaluation of the usefulness of the factors and metrics, and the DMAIC framework as a whole, to improve Agile/DevOps team performance, is currently ongoing.

The set-up of this research has important limitations. It is quite possible that a bias results from selecting interviewees from one organization, namely GapGemini. This risk is reduced, because we selected interviewees who manage teams in different client organizations. Furthermore, there are big differences between the situations of the interviewees. They had taken different approaches to adopt Agile/DevOps, and each struggled with different issues. It seems that no guiding structure was provided to establish Agile/DevOps at the different clients. These limitations mean that the framework should be seen as a series of hypotheses.

Measuring the performance of teams goes against the spirit of autonomy that characterizes Agile methods. This may hinder adoption. On the other hand, there is a clear need for guidance and alignment, especially when teams are managed remotely. Further research must point out which metrics will be most popular to be adopted by the teams themselves, which metrics must be obliged, and which metrics must be made part of the CI/CD pipeline tooling.

References

[1] M. Lindvall, V. Basili, B. Boehm, P. Costa, K. Dangle, F. Shull, R. Tesoriero, L. Williams, M. Zelkowitz, Empirical findings in agile methods, in: Extreme Programming and Agile Methods, LNCS 2418, 2002, pp. 197–207.

[2] A. Alliance, Manifesto for Agile software development, agilemanifesto.org (2001).

[3] R. Vidgen, X. Wang, Coevolving Systems and the Organization of Agile Software Development, Information Systems Research 20 (3) (2009) 355–376.

[4] J. Livermore, Factors that impact implementing an agile software development methodology, Proceedings 2007 IEEE SoutheastCon (2007).

[5] R. de Feijter, S. Overbeek, R. van Vliet, E. Jagroep, S. Brinkkemper, DevOps Competences and Maturity for Software Producing Organizations, in: Enterprise, Business-Process and Information Systems Modeling, 2018, pp. 244–259.

[6] L. Leite, C. Rocha, F. Kon, D. Milojicic, P. Meirelles, A Survey of DevOps Concepts and Challenges, ACM Computing Surveys 52 (6) (2020) 1–35.

[7] C. Ebert, G. Gallardo, J. Hernantes, N. Serrano, DevOps, IEEE Software 33 (3) (2016) 94–100.

[8] L. Zhu, L. Bass, G. Champlin-Scharff, DevOps and its Practices, IEEE Software 33 (3) (2016) 32–34.

[9] S. Faraj, L. Sproull, Coordinating Expertise in Software Development Teams, Management Science 46 (12) (2000) 1554–1568.

[10] T. Dingsøyr, T. E. Faegri, T. Dybå, B. Haugset, Y. Lindsjorn, Team Performance in Software Development: Research Results versus Agile Principles, IEEE Software 33 (4) (2016) 106–110.

[11] T. Visser, Towards the continuous improvement of Agile/DevOps team performance, Master's thesis, Tilburg School of Economics and Management (2022).

[12] R. Wieringa, Design Science Methodology for Information Systems and Software Engineering, Springer, 2014.

[13] A. Prashar, Adoption of Six Sigma DMAIC to reduce cost of poor quality, International Journal of Productivity and Performance Management 63 (1) (2014) 103–126.

[14] G. D. Maayan, 6 Great DevOps Metrics - and How to Choose the Right Metrics (2 2021).

[15] T. Hall, DevOps metrics (9 2020).
URL https://www.atlassian.com/devops/frameworks/devops-metrics

[16] Digital.ai, 15th State of Agile Report: Agile adoption accelerates across the enterprise, Tech. rep. (2021).

[17] M. Jørgensen, A Critique of How We Measure and Interpret the Accuracy of Software Development Effort Estimation , 1st International Workshop on Software Productivity Analysis and Cost Estimation (2007) 15–22.

[18] B. Snyder, B. Curtis, Using Analytics to Guide Improvement during an Agile–DevOps Transformation, IEEE Software 35 (1) (2018) 78–83. doi:10.1109/ms.2017.4541032.

[19] M. Sokovic, D. Pavletic, K. Kern Pipan, Quality Improvement Methodologies – PDCA Cycle, RADAR Matrix, DMAIC and DFSS, Journal of Achievements in materials and manufacturing engineering 43 (1) (2010) 476–483.