Master’s thesis:

Optimizing development performance through team composition and team culture factors in modern software development organizations

Author:
Igor Andriushchenko

Thesis supervisor:
Viroj Jienwatcharamongkhol

Submitted on
20th October 2019
Acknowledgements

Thanks to my tutor Viroj Jienwatcharamongkhol for his valuable contributions and opinions on the thesis, thorough reading and always curious mind.

Big shout out to Martin Svensson for organizing a great course on Research Methodologies that became a starting point for the idea of this research.

Huge thanks to all research participants from around the globe, it would have never happened without you! Thanks for your time spent on filling in the survey form and agreeing to answer all the tricky questions.

Token of my appreciation to Snow Software, my employer, and namely my colleagues Martin Roth and Victoria Normark who were the first to hear about the model, hypothesis and asked very important questions at the earliest stage possible – all according to the shift-left and DevOps principles!

As always, thanks to my caring partner, best friend and amazing wife, Masha. I love you to the Moon and back. And will keep writing embarrassing acknowledgements to you, take my word.
Abstract

Modern organizations that follow Agile development principles and practice DevOps aim at maximizing software release frequency while minimizing the number of defects associated with them. To achieve this goal, the companies perform the organizational transformation that is associated with significant costs and time investments. The key to success is building high-performing development teams. The existing literature does not extensively cover development performance optimization in DevOps organizations that are in the process of transformation. It provides limited information for the organizations willing to speed up the process and remove the pain points. As the result, organizations suffer economic losses from adjusting the development practices the transformation process, taking the new technology to the customers takes longer time comparing to the high performing organizations, which can be seen as losses for the modern highly digitalized economy.

This study explores whether the team composition, team culture and other organizational factors influence performance of development teams by surveying high-performant organizations that completed the transformation, and fitting a statistical model the collected data. The model validates assumptions and serves as a useful tool for low-performing organizations to adjust their team composition and culture accordingly. The novel metrics for assessing the development team performance in the DevOps context is proposed.

The research concludes that teams that are mainly composed of developers and have access to shared expertise of principal DevOps and Security engineers placed outside of the team, perform the best. Autonomy of developers within the team is another significant factor for achieving the optimal performance. This confirms findings about individual autonomy of the classic studies and places them into the modern DevOps context. The study shows that there exists no direct relationship between the number of quality engineers and quality which may indicate a turn in the classic QA theory that assumes QA engineers as integral player in organizational quality. Finally, it estimates an optimal rate of developers and non-developers in a team for the highest performance and demonstrates that high performance can be achieved by organizations regardless of sizes, product types and modern development methodologies.
Table of content

Acknowledgements .............................................................................................................. 2
Abstract ............................................................................................................................... 3
Table of content ................................................................................................................... 4
Lists of Figures and Tables ................................................................................................. 6
Chapter 1 Introduction ............................................................................................................ 8
1.1 Problem discussion and formulation .............................................................................. 10
1.2 Delimitations .................................................................................................................. 13
1.3 Thesis Structure ............................................................................................................. 14
Chapter 2 Theory .............................................................................................................. 15
2.1 Agile, SCRUM and other development methodologies ................................................... 15
2.2 DevOps – definition and importance in the context of Agile methods ......................... 15
2.3 Software team performance metrics evolution ............................................................... 18
2.4 Team composition and culture research ...................................................................... 19
2.5 Expertise composition research ................................................................................... 20
2.6 Theory summary ......................................................................................................... 21
Chapter 3 Methodology ......................................................................................................... 23
3.1 Methods overview ........................................................................................................ 23
3.2 Proposed measures ....................................................................................................... 24
3.3 Statistical modelling ..................................................................................................... 27
  3.3.4 Model variables ...................................................................................................... 27
  3.3.5 Model equation ...................................................................................................... 28
3.4 Sample selection ........................................................................................................... 30
3.5 Survey design .............................................................................................................. 30
Chapter 4 Descriptive statistics and model reliability checks .................................................... 32
4.1 Exploring visible relations between response and independent variables ..................... 37
4.2 Checking Linear Model assumptions ............................................................................ 39
  4.2.1 Response Variable Distribution ............................................................................... 39
  4.2.2 Assessing non-collinearity between independent variables ........................................ 42
  4.2.3 Linear relation between independent variables and response variable ................. 44
4.2 Fitting and validating the polynomial model .................................................................. 45
4.3 Statistical modeling results ........................................................................................... 47
  4.3.1 Testing additional variables .................................................................................... 51
4.4 Analysis summary ........................................................................................................ 53
Chapter 5 Conclusions ........................................................................................................... 55
5.1 Implications and contributions of the study ................................................................. 56
5.2 Study limitations and future directions of research ...................................................... 58
References ...................................................................................................................... 60
Appendix A: Survey design ............................................................................................... 66
Appendix B: Statistical modeling details ........................................................................... 71
   ANOVA of Models 1-4 .................................................................................................. 71
   Comparing polynomial models ................................................................................... 71
   Fitting Model 1 without point 31 ............................................................................... 72
   GVLMA check for Model 1 ....................................................................................... 73
   Testing additional variables with Model 1 ................................................................. 73
   Final model - adding RQa to the equation ................................................................. 74
   Model with only original variables without composite measurements ..................... 75
Lists of Figures and Tables

Figure 4.1 Distribution of eligible and non-eligible responses to survey ........................................ 32
Figure 4.2 Location of responding team ............................................................................................. 32
Figure 4.3 Distribution of product type among responders to the survey ............................................ 33
Figure 4.4 Distribution of company sizes among responders to the survey ........................................ 33
Figure 4.5 Distribution of development method type between responding teams ............................... 34
Figure 4.6 Descriptive statistics boxplots of main variables ................................................................. 35
Figure 4.7 Relation plots of rate of engineers to main response variables ......................................... 37
Figure 4.8 Relation plots of rate of QA to main response variables .................................................... 37
Figure 4.9 Relation plots of rate of QA to main response variables .................................................... 38
Figure 4.10 Relationship between Performance and fixed development factors ................................. 38
Figure 4.11 Distribution and density of the response variable ............................................................. 39
Figure 4.12 Distribution and density of log-transformed response variable ........................................ 40
Figure 4.13 Distribution and density of square root-transformed response variable ............................ 41
Figure 4.14 Boxplots of P and square root-transformed P ................................................................. 42
Figure 4.15 Correlation heatmaps for the initial variable set ............................................................... 43
Figure 4.16 Correlation heatmaps for different independent variable sets .......................................... 43
Figure 4.17 Linearity plots for independent variables ......................................................................... 44
Figure 4.18 Q-Q plot for residuals of Model 1 .................................................................................... 47
Figure 4.19 Scale-Location residuals vs fitted values plot for Model 1 ................................................. 48
Figure 4.20 Q-Q plot for residuals of Model 1 after removing the outlier ........................................... 49
Figure 4.21 Scale-Location residuals vs fitted values plot for Model 1 after removing the outlier ....... 49
Figure 4.22 Residuals vs Fitted plot for Model 1 after removing the outlier ........................................ 50
Figure 4.23 Residuals vs Fitted plot for Model 1 after removing the outlier ........................................ 50
Figure 4.24 Model with RQa, residuals checks .................................................................................... 53

Table 4.1 Descriptive statistics values for main variables .................................................................. 34
Table 4.2 Polynomial model R-squared comparison .......................................................................... 45
Table 4.3 ANOVA of four best performing models ............................................................................. 45
Table 4.4 Model fit summary statistics for Model 1 and Model 2 ....................................................... 47
Table 4.5 Comparison of models with and without the outlier ......................................................... 48
Table 4.6 Polynomial regression Model 1 coefficients after removing the outlier ............................. 51
Table 4.7 Model comparison after adding RQa interaction ................................................................. 52
Table 4.8 Final model statistics after adding RQa interaction ............................................................. 53
Table 5.1 ANOVA of four best performing models ............................................................................. 71
Table 5.2 Polynomial regression Model 1 statistical summary ......................................................... 71
Table 5.3 Polynomial regression Model 2 statistical summary .......................................................... 72
Table 5.4 Model 1 Variance Inflation Factor parameters .................................................................... 72
Table 5.5 Model 2 Variance Inflation Factor parameters .................................................................... 72
Table 5.6 Polynomial regression Model 1 statistical summary after removing the outlier ................. 72
Table 5.7 Assessment of general statistical acceptance parameters for Model 1 ............................... 73
Table 5.8 Statistics summary of Model 1 with added LastHire variable ............................................ 73
Table 5.9 Statistics summary of Model 1 with added NT variable .................................................... 73
Table 5.10 Statistics summary of Model 1 with added DevMethod variable ...................................... 73
Table 5.11 Statistics summary of Model 1 with added RQa ............................................................... 74
Table 5.12 Assessment of statistical acceptance parameters for Model with RQa ............................ 74
Table 5.13 VIF calculation for the final Model with RQa............................................................. 74
Table 5.14 Statistics of the non-composite variables model .......................................................... 75
Chapter 1 Introduction

Nowadays, IT product organizations seek for maximizing the development performance without sacrificing quality and enabling fast feedback loops from customers - for quick response to customer and market needs. This allows to achieve high performance in delivery and minimize the lead time for the idea to reach the customer – the core principle of DevOps set of methodologies and practices (Humble & Kim, 2018). It was demonstrated that software organizations that practice DevOps and deliver software products quickly and reliably achieve better overall organizational performance and higher customer success rates (Forsgren, Humble, & Kim, 2019).

These organizations use a variety of software development practices based on Agile development methodology, built on principles of Lean Management and extended with the toolbox of the DevOps methodology. The transformation, however, takes a significant toll on organizational resources and productivity, requires an executive buy-in, leads to introduction of new process, roles and principles. The tight integration of software products into the modern economy leads to economic losses, lost revenues and slower GDP growth rates due to organizations struggling with the digital transformation (Park & Roome, 2017).

According to the latest figures, only 20% of the software product organizations are rated as high-performing with regards to their DevOps, delivery and release practices (Forsgren et al., 2019). Helping the remaining 80% of the organizations that are in earlier stages of their transformation paths to optimize the software development routines and enable high-performance in delivery and quality, would benefit not only to these organizations but also local and global economies, overall technology advancement – making it a valid and important area for scientific research.

Achieving the fast delivery rate with a stable high quality is a non-trivial problem - the organizations often experiment with team structures and development methodologies to find the best performing methods (Erich, Amrit, & Daneva, 2017). This thesis researches the influence of team composition, team culture and various organizational factors to maximize a novel joint metric of release and quality through building a statistical model, based on the collected survey data from modern mature Agile development organizations practicing DevOps belonging to different types, sizes and industries.

In general, the Agile methodology can be applied to the development process using different methodologies – such as SCRUM (Schwaber, 1997); XP (Extreme Programming) (Beck & Gamma, 2000); Kanban (Anderson, Concas, Lunesu, & Marchesi, 2011) and others.

In the recent years, DevOps principles were introduced into the common development practices and enhanced them with new set of cultural, tooling and organizational transformations. It led to significant changes in software engineering by giving more freedom and self-sufficiency to engineers (Humble & Kim, 2018). This transformation caused the introduction of multi-profile roles to the team, as the opposite to the specialists, – for example, Full-Stack developer partially replaced specialized Backend and Frontend developers; quality assurance function became the developer’s responsibility. Furthermore, the developers were tasked with Security, Build and Integration, Site Reliability and even product ownership responsibilities as the addition to their main role.

One of the important questions for the organizations in the middle of their DevOps transformation journeys – which has the main goal of removing the bottlenecks of engineering performance and minimizing the lead time for the new features to customer – would be how to organize teams for a
better performance, which roles to keep and which to merge into the developers’ responsibilities. This transformation can be seen as the key enabler of high performance for development and higher productivity for the entire digital economy (Humble & Kim, 2018).

Organizations approached the development paradigm change to different extent – some completely eliminated specialized roles while the other still kept the senior expertise of these roles while giving simpler tasks to the developers (Feitelson, Frachtenberg, & Beck, 2013). Recent research shows that high-performing IT product organizations have completed the DevOps transformation at both the team and organizational scale (Humble & Kim, 2018). At the same time, many companies are still struggling with their DevOps transformation and can’t achieve high-performance (Erich, Amrit, & Daneva, 2017). Due to DevOps being a relatively new, the scientific research has relatively better coverage of generalistic Agile and SCRUM methodologies and corresponding project and team management practices. At the same time, with DevOps focusing on expertise integration, automating processes and numerically measuring performance in order to achieve the lowest product-to-market time (Debois, 2011). Based on this approach, it became sensible to capture the organizational performance as the relationship between release frequency and quality.

Developing a theory behind the team and expertise distribution for transforming organizations into high-performing entities that utilize DevOps can enable quicker and smoother transition and help avoid losses related to underperforming teams losses, hence, build and deliver the product quicker, enable new technology and drive the economy. This research would help build the basic scientific understanding of what are the main contributing factors to high-performing teams in the DevOps-practicing organization, finding how the team composition, team culture and organizational factors affect the development performance.

From the practical point of view, the resulting statistical model could be directly applied to organizational practices for optimizing software development performance rates through expertise distribution and team culture adjustment.

This study can be used as the initial point in exploring the specialized competences role shifts and culture shifts associated with the final stages of the DevOps transformation.
1.1 Problem discussion and formulation

This study aims at researching and solving the problem of the development performance optimization via assessing professional engineering team composition, team culture prerequisites and other organizational factors in modern companies that achieved high maturity level in practicing DevOps as part of the Agile development. These organizations have been successful in their transformation through repeated trial and error process, experimenting with various development process variable, and possess a significant amount of practical knowledge that has not been verified or generalized from the theoretical or scientific standpoints to provide tools and pathways to less mature organizations that are starting or started their transformation journeys (Wiedemann, Forsgren, Wiesche, Gewald, & Krcmar, 2019).

Solving this problem would significantly decrease organizational losses associated with digital transformation in software development companies aimed at decreasing the lead time of development for value from idea to the customer, the key concept of DevOps. This transformation is the key for most of the organizations to achieving performance levels comparable to the technology leaders such as Facebook, Google, Apple and boosting the modern highly-digitalized economy (Feitelson, Frachtenberg, & Beck, 2013).

This study focuses on finding whether there exists a statistically significant relationship between the software development organization performance and the blend of expertise composition, team culture, organization size, the product type and other factors that are associated with software development in the organizations. The research explores a relatively new domain of DevOps development practices and is specific to mature organizations that has reached the high rates of release frequency and minimized the lead time for the value to reach the customer.

The theory applicable to the topic is adjacent to Agile development methodologies research. Agile was created as a response to the cumbersome and inflexible Waterfall development (Highsmith & Cockburn, 2001). It was followed by the adoption of already existing SCRUM process and extending Agile with a practical approach to conducting team work and collaboration (Schwaber & Beedle, 2002). Existing research of organizational performance focuses on Agile and SCRUM project management and high-level transformation practices. More granular view into how teams are built and work, and what is enabling the high-performance throughput coupled with maximizing quality, is missing from the existing research. This can be explained by irregularities in Agile- and SCRUM-companies structure, organizational composition and established processes. At first, the practices were relatively new to the field, and different organizations approached it differently with no reference or golden standard of a successful transformation. Secondly, the performance and outputs of such organizations were distributed unevenly. It caused challenges in measuring the development performance and finding a metric that would fairly reflect the performance of a particular team and its quality.

DevOps methodology re-iterated on existing Agile and SCRUM practices (Lwakatare, Kuvaja, & Oivo, 2016), without replacing them but rather enhancing with changes to processes and expertise composition. The DevOps definition is highly varies between the papers and practitioners (Erich, Amrit, & Daneva, 2017). Its definition spectrum starts with “automating everything” and being a culture that promotes cross-team collaboration. This research takes the practitioners’ definition of DevOps – it’s the set of practices that helps organizations to make path of the product and its
features from the idea to the customer as short as possible, include direct feedback from the
customer and built-in the experimentation and collaboration into the organizational culture\(^1\).

Adoption of DevOps led to increased automation of various processes in development and required
making faster decisions – in this new setting the old approach of maintaining separate teams of
developers and test engineers led to delayed releases and disconnect between the expertise domains
(Feitelson et al., 2013). This situation inspired organizations to seek for ways to employ multi-
skilled engineers that would handle most of the development on their own and possess the core
expertise of multiple roles (Erich et al., 2017).

Due to the lack of the practical and theoretical evidence on how different levels of expertise
ownership affect development performance, organizations approached this challenge differently.
The resulting expertise composition seemed to vary greatly between organizations (Pransomchinske,
2019). The statistical generalization of how high-performing organizations approach this problem,
could provide useful theoretical and practical insights into building the best-performing teams in
DevOps-practicing organizations.

Human resources team composition and software performance metrics research is another
important source of existing knowledge that can be utilized for understanding the factors that affect
development team performance.

A number of papers focused on finding optimal team composition from the personality (Gilal,
Jaafar, Omar, Basri, & Waqas, 2016) or diversity (Liang, Liu, Lin, & Lin, 2007) perspective for
particular roles in development. At the same time, the expertise composition of teams is not
covered by the research efforts. This may be explained by the way the developers were seen before
Agile – as highly specialized engineers that contribute to their unique domain expertise areas and
cannot be measured in terms of more generic roles. It was not possible to imagine back in 2000-s
that one developer can perform testing, security analysis, deployment and operations while being
efficient in the creating the value for the product at the same time (Humble & Kim, 2018).

Different measures for team performance were proposed over the last decades. Output quality was
measured by number of post-delivery defects; adherence - whether the project is delivered on time
(Huckman, Staats, & Upton, 2009). Other researchers create complex metrics in which they
capture both quality of the product and also user satisfaction, meeting a delivery schedule and
staying within project budget (Jiang, Klein, Hwang, Huang, & Hung, 2004).

The mentioned measures are rather specific to a traditional project lifecycle that always includes
well-defined requirements to the product, a clear delivery phase which needs to be completed on
time and no continued development after the product was released. In DevOps settings, deliveries
are performed frequently by design, and the main characteristic of a well-performing organization
is how frequently it ships releases to satisfy customer demands. Essentially, the customers receive
a Product-as-a-Service which is being continuously re-shaped and enhanced to meet changing
market and customers’ needs. In this setting, the requirements are elusive, the constant dialogue
with the customers is essential, experimentation and idea validation become the part of the product
(Jabbari, bin Ali, Petersen, & Tanveer, 2016). To reflect this change, the research introduces its
own measure that takes into account previous attempts to capture the quality but is better suited
for DevOps continuous delivery specifics.

\(^1\) Donovan Brown | What is DevOps? (2015). Retrieved February 17, 2019, from
http://donovanbrown.com/post/what-is-devops
This study theoretical contribution generalizes existing knowledge in areas of software
development performance and quality estimation, team composition, team and organizational
culture in different variations of Agile development methodologies.

The study applies and extends the existing theory to answer the research question: whether there
exists a significant relationship between the development performance (expressed in terms of
release frequency and defect counts) and such variables as team professional roles composition,
team culture and organizational factors of highly performant product-developing IT companies
across the world.
1.2 Delimitations

This study focuses on assessing established software development organizations that are not being in the middle of an organizational transformation. The sample selection is limited to DevOps-practicing software development organizations that produce software products and have reached a high maturity in their DevOps transformation. The products produced by these organizations are considered among the best-in-class or well. The developed models and measures are not applicable to DevOps teams or organizations that have not used or have not completed the DevOps transformation. The study does acknowledge but does not attempt to capture all different factors of complex interactions that any engineering team is involved with. Instead, it picks up the minimal viable number of variables reviewed by existing scientific contributions - team composition, culture and organizational fixed factors, and explores their influence on development performance.
1.3 Thesis Structure

The “Theory” chapter describes existing theories in the field and draws the connection between them and proposed research questions.

The “Methodology” chapter describes chosen methodology, study design, data collection and analysis techniques.

The “Analysis” chapter reports the data collection results, descriptive statistics of the data set, describes statistical analysis.

“Conclusions” summarizes the research, discusses the results in the broader context and sets the direction for the future research.

“Appendix A” section contains the example of the survey sent to the respondents.

“Appendix B” section contains additional results of statistical modeling that were found to be too extensive to be included in the main body of text but nevertheless significant for a curious reader and providing useful insights into the statistical modeling process.
Chapter 2 Theory

2.1 Agile, SCRUM and other development methodologies

Agile methodology is at the core of various modern software methodologies and frameworks (Beck et al., 2001). It emerged as the reaction to unsatisfying performance of existing waterfall-like development methodologies that failed to deliver software in time and make it satisfy customer's expectations (Ramamoorthy, Tsai, Yamaura, & Bhide, 1985). Agile manifesto set “individuals and interactions over processes and tools, working software over comprehensive documentation, customer collaboration over contract negotiation, responding to change over following a plan” (Beck et al., 2001). Existing frameworks such as SCRUM were found to be applicable and useful under the new methodology, Agile Scrum development remains actual nowadays (Schwaber, 1997). Adoption of Agile development started in 2000-s and until mid-2000-s it was not followed by other departments and functions of companies such as infrastructure and operations (Debois, 2008). Before the transformation, this department remained rigid and inflexible causing Agile projects to fail to deliver on the expectations due to the lack of communication between development and operations, long lead times for new products to be available for the customers, fragile and error-prone infrastructure, lack of mechanisms for getting feedback from the customers (Elbanna & Sarker, 2016).

Other popular Agile-based methods are Crystal Clear, Dynamic Systems Development Method, Feature-driven Development (FDD), Test-driven Development (TDD), Kanban.

The challenge of scaling Agile methods to the company led to developing of scaling methodologies such as Discipline Agile Delivery (DAD), Large Scale Scrum (LeSS), Scaled Agile Framework (SAFe), “Spotify”, Nexus, Recipes for Agile Governance in the Enterprise (RAGE) (Alqudah & Razali, 2016).

The DevOps a set of practices is mentioned as an integral part of SAFe and Spotify frameworks can be applied to any of these frameworks and methodologies. According to the latest research it is names as one of the key transformations on the way to high performance in delivery (Humble & Kim, 2018)

2.2 DevOps – definition and importance in the context of Agile methods

What exactly is DevOps – this question has no definitive answer, since both academia and practitioners have different definitions of DevOps which range from the simple cooperation between development and operations to the new development framework (Erich et al., 2017). Historical perspective to the DevOps adoption and development can provide understanding of the phenomenon and its modern context.

The beginnings of DevOps adoption are rooted in the non-academical environment of VisibleOps and its satellite conferences for practitioners. The VisibleOps concept gained traction in mid-2000s and was making the process of service and infrastructure management simpler (Behr, Kim, & Spafford, 2007). The VisibleOps advocated for managing IT systems in a new way with implementing a simplified version of ITIL governance framework (Cervone, 2008). It was done in order to achieve more flexibility and visibility into IT operations, and also break down the lock on big system vendors that required a constant presence and work of trained professionals for performing even a simple operation. Smaller vendors started to offer specific solutions that could co-exist with big vendor products but made one or another aspect of running them much simpler.
A number of software conferences were held and focused on improving the delivery speed and practices and utilizing the new paradigm started by VisibleOps.

The newly established Velocity conference played a big role in accelerating the changes. In 2008 and 2009, it showcased new tools such as Puppet and Chef that will soon become the standard for running Infrastructure-as-Code (IaC) and allow for complete automation of the entire company infrastructure provisioning. The most notable talk of the conference was “10+ deploys per day: Dev and Ops cooperation at Flickr”\(^2\), a revolutionary insight for its time when even the most successful companies had a successful delivery rate at about 10 releases per year. The presentation covered the use of IaC, empowering developers and building the culture that supports quick and robust releases. Later in 2009, system administration practitioners Debois and Shafer, inspired by the Dev and Ops talk and conversations held during the Velocity conference, organized the conference DevOpsDays in Belgium. They create a short hashtag for the conference: #DevOps\(^3\). From the very beginning, there was no clear or “classic” definition of DevOps – it was represented by a set of activities that aimed to change the existing situation in software development that was caused by cumbersome and inflexible operations practices.

During the DevOpsDays conference, DevOps received a name, tools and approach for implementing changes – it started a big movement that unknowingly led to making software companies and their delivery more efficient and relevant to the customer. Due to a rapid growth, the term quickly became a buzzword that meant various concepts and approaches (Lwakatare, Kuvaja, & Oivo, 2015). When reviewing the historical changes in the DevOps term meaning, it is noticeable that in 2009 DevOps started with a simple technical goal – to bring Agile principles into system administration\(^3\). After the initial adoption, it became more specific and proposed to achieve a better collaboration between development and operations department of companies and breakdown the abstract wall between them (Hüttermann, 2012). Within the next two years, the transformation went even further and encircled the design, development, operations and management practices. The DevOps practitioner and book author Jez Humble defines it as “a cross-disciplinary community of practice dedicated to the study of building, evolving and operating rapidly-changing resilient systems at scale”\(^4\).

Ernest Mueller, one of the authors of the popular blog “Agile Admin” proposes another definition: “DevOps is the practice of operations and development engineers participating together in the entire service lifecycle, from design through the development process to production support.” \(^4\)

The academical definition of DevOps is also varying between different papers and researchers. A recent meta-study of the DevOps literature synthesized a collective definition of DevOps based on the most common definitions found in publications (Jabbari et al., 2016). The final definition was expressed as following: “DevOps is a development methodology aimed at bridging the gap between Development (Dev) and Operations, emphasizing communication and collaboration, continuous integration, quality assurance and delivery with automated deployment utilizing a set of development practices.”

---

\(^2\) 10+ Deploys Per Day: Dev and Ops Cooperation at Flickr: Velocity 2009 - O’Reilly Conferences, June 22 - 24, 2009 - San Jose, CA
Importantly, this definition puts DevOps in the same line as Agile, SCRUM, XP, SAFe and other development methodologies and frameworks. Over time, DevOps evolved from being an agile system administration concept to a development methodology that can enhance leading development practices and enable high-performance delivery.

What is DevOps in the context of this thesis? We would propose to unite the abovementioned definition from Jabbari with a softer formulation made by Donovan Brown, a DevOps practitioner from Microsoft and a popular blog author. He describes it as “the union of people, process, and products to enable continuous delivery of value to our end users” ⁵. This definition later became one of the most used in the industry. Here, the crucial difference is that the academical definition focuses on the technical aspects of DevOps within an organization, while practitioners look beyond it and add also customer success to the picture. By doing this, the practitioners turn DevOps from internal into also external initiative and build a set of processes that support this orientation. Feedback from the customer is seen as essential component of continuous improvement, and organization should implement various channels for receiving it – through project managers and owners, product telemetry and direct user feedback. Therefore, the proposed definition to use here and later:

“DevOps is a development methodology aimed at enabling continuous delivery of value to end users by bridging the gap between Development (Dev) and Operations, emphasizing communication and collaboration culture, continuous integration, quality assurance and delivery with automated processes”.

When talking about DevOps as a development methodology, it is worth mentioning that researchers generally agree that DevOps extends Agile (De Bayser, Azevedo, & Cerqueira, 2015). Mature Agile adoption is viewed employing an Agile development methodology as a prerequisite, support or enabler to kicking-off DevOps initiative (Lwakatare et al., 2016), (Bass, Weber, & Zhu, 2015). SCRUM, SAFe, LeSS and other Agile-based frameworks are highly compatible with DevOps and can be augmented without significantly disrupting the existing development flow (Alqudah & Razali, 2016). From the thesis point of view, DevOps is the specific way a software organization works, no matter which Agile framework it works according to.

Despite DevOps becoming a standalone development methodology, the academic research still lacks evidence of DevOps efficiency (Erich et al., 2017). The practitioners conducted interviews with various organizations that develop software and concluded that top-performers have mature DevOps practices while low-performers have just started DevOps implementation and are currently in the process of transformation (Humble & Kim, 2018).

The most recent empirical review of how DevOps affects the organizational performance demonstrated that only 1 out of 5 software organizations reached high level of DevOps maturity. At the same time, the benefits of the approach yield only at the latest stage leaving those who are in early or medium stages of DevOps maturity far behind with very moderate performance gains (Forsgren et al., 2019). Once organizations reach the high maturity in DevOps, they become capable of improving overall organizational performance including financial, customer satisfaction and other areas and are twice as likely to reach or exceed their goals.

---

2.3 Software team performance metrics evolution

Measuring performance of software engineering has been evolving together with development practices. In the pre-DevOps world, it was mainly based on such factors as number of defects, meeting the project schedule and the amount of work required to achieve a target level of customer satisfaction with the released product (Edmondson & Nembhard, 2009).

Another popular research metrics for development team effectiveness and efficiency was performing self-rated team efficiency survey which was based on team members answers to the question whether the team works efficiently or not, as perceived or felt by the respondent (Pearce & Sims Jr, 2002).

A different aspect of waterfall development was captured by the requirement fit metrics - it reflected how well released and delivered software corresponds to the initial requirements agreed upon with a customer (Liang et al., 2007).

These measures were not applicable to flexible, data-driven and often not maintaining formal requirements organization that practice DevOps principles. They had no fixed release dates, project schedules and fixed requirements that need to be satisfied. Continuous releases with multiple deploys per days made the requirements obsolete and the traditional performance measures counter-productive6.

The new Agile-age development KPIs and metrics were inspired and adopted by Product Management offices, they introduced velocity (how many atomic work items completed during a period of time), burndown rate (how well is team performing against the estimate of closed items by the end of a fixed period), lead time to implementing a work item from idea to production and similar metrics (Kupiainen, Mäntylä, & Itkonen, 2015).

In the recent years, mature DevOps organizations were also seen to heavily invest in maximizing a number of releases per day and automating most of the testing activities to improve the relative quality of frequent releases (Humble & Kim, 2018). These changes required not only team but also organizational-wide changes and executive management commitment to building the high performing organization.

This focus introduced another dimension of the performance metrics – the relative quality of the product. The term quality itself is similar to DevOps in having various meanings, and there are very different views on its content. Quality according to the ISO 9000 standard contributor Crosby, is when a product is conformant to requirements and has zero defects (Crosby, 1985). This definition was later overthrown by software engineering industry that argued that there is always a tradeoff between the product cost and acceptable number of non-critical defects. The classic paper by Kitchenham discusses various view on what is meant by software quality (B. Kitchenham & Pfleeger, 1996). It offers a set of views that quality is perceived from – such as transcendental or philosophical, user view, manufacturer view, product view and value-based view. The last category is pointing to the software quality view that would become the motivation behind Agile methodology, DevOps and modern software development practices. Kitchenham discusses certain benefits in using more agile “value-based” view on quality that would meet both customer demand and engineering capacities. Nevertheless, in 90-s, product and manufacturer’s views were dominant and utilized defect counts and rework costs as main quality metrics. A popular metric was Defect

---

6 10+ Deploys Per Day: Dev and Ops Cooperation at Flickr: Velocity 2009 - O’Reilly Conferences, June 22 - 24, 2009 - San Jose, CA
Density – a number of known defects divided by product size (in lines of code) and system portability which indicated how complex would it be to move the existing system to the target environment comparing to creating a new system for the target environment (Fenton & Bieman, 2014).

In 2000-s with the initial adoption of the Agile development, the need to review the existing approach to testing and quality assurance has emerged. McBreen first talks about agile quality assurance – a process in which software development can respond to change when the customer requires the change (McBreen, 2003). In this context, the quality measure that is based on defects numbers only becomes inapplicable.

The further adoption of Agile practices and automation led to the introduction of new metrics that attempt to go beyond lines of code and number of defects as the main measure of quality. Release readiness index is the rate of defects to the current defect removal rate in the project subtracted by the difference between the rate between executed and passed tests (Staron, Meding, & Palm, 2012). This existence of this approach demonstrates inapplicability of “zero-defects” quality practices to the modern Agile development and the focus on getting product to the market with an acceptable level of quality assessed mainly by automation. This can be viewed as one of the early attempts in measuring quality that can be applied to the DevOps organizations.

Research of Perera et al. makes a relatively rare attempt to capture the software quality in DevOps organizations, however, it utilizes relatively old and rigid ISO 9126 standard to build a metric based on Functionality, Reliability, Usability, Efficiency, Maintainability and Portability (Perera, Silva, & Perera, 2017).

As with the DevOps engineering management, the DevOps quality assurance research has no established quality metrics that can capture not only a classic defect-based quality dimension but also add release frequency as an important metric for all DevOps organizations – since the release rate indicates how quick changes can be delivered to the customer, and how quick the customer can receive additional value from the product. The closest metric is mentioned by Hoegl et al in a pre-Agile paper – in it, the team performance is measured as the extent to which the team is meeting established quality and cost and time objectives of the project (Hoegl & Gemuenden, 2001).

The most recent “State of DevOps 2019” study measured DevOps performance among organizations divided into clusters (Elite, High, Medium, Low) and shown that the release frequency degrades from multiple deploys per day for elite performers to every six month in Low performers (Forsgren et al., 2019). This metric is not continuous in the study and only indicates the performance bracket. The same study also measures Change failure rate as which percentage of changes in production associated with worsened service performance or issues – it includes infrastructure and operations problems.

The above-mentioned quality and release performance metrics can be adjusted and adopted to the DevOps development context by removing the project from the context and maximizing the importance of frequent releases with maximized quality. A new team performance metric is designed for the purpose of the research and proposed in Chapter 3.2.

2.4 Team composition and culture research

The topic of team composition in Agile teams has significant theoretical and practical contribution to it. The concept of fluid team boundaries in Agile development experiments with team composition and introduces flexible and interchangeable roles shared or transferred between teams.
based on the need (Edmondson & Nembhard, 2009). Based on this concept, Huckman and Staats research how to achieve the project best performance based on a team composition defined by various factors such as individual experience level, team familiarity, project complexity, budget and duration (Huckman et al., 2009). It is a pre-DevOps study and it has signs of waterfall delivery – projects have fixed outcomes, delivery times, and team performance is measured based on meeting the project schedule and goal. It is not the case with DevOps, where project releases happen daily, and there is no fixed duration for a particular project. Nevertheless, it provides a useful insight into how more flexible teams can contribute to better project outcomes and can be used as a starting point in researching what makes teams perform better in DevOps organizations.

Team diversity was proven to be important to the project performance when it comes to expanding the knowledge diversity (Liang et al., 2007). The more diverse team knowledge is, the better outcomes a project may expect to achieve, and hence the better performance it may show at the end of the project life cycle.

Autonomy climate within the team and encouraging experimentation and individual independence was shown as positively affecting knowledge integration (Basaglia, Caporarello, Magni, & Pennarola, 2010), which in turn may lead to better development effectiveness and efficiency. For assessing the team autonomy, the researchers often use the five-item scale developed by Langfred and used in interviews or surveys of team members (Langfred, 2005).

Personality types of co-workers and its effect on performance were studied by Gorla and Lam, which demonstrated that particular types of engineers perform better – in most cases, those who have rather extravert personality and are better at communication (Gorla & Lam, 2004).

More recent studies focus more on external factors such as team and organizational culture, managerial buy-in, overall customer focus of the organization, existence of team recognition and transparent feedback - as enablers of DevOps transformation and adoption, rather than attempts to measure performance as a function of particular professional trait (Kamuto & Langerman, 2017). The team diversity, fluidity and suitable personality type composition discussed in pre-DevOps literature can be seen as a pre-requisite to a successful DevOps transformation.

The Agile team performance theory studies teamwork quality of the team from the psychological and social sciences perspectives – in it, researchers look at the teamwork as a social phenomenon. The team performance is measured as a function of Shared Mental Models and Backup Behavior (Schmidt, Kude, Heinzl, & Mithas, 2014). Other researchers also include functions of Team Leadership, Mutual Performance Monitoring, Team orientation and Adaptability (Dingsøyr & Dybå, 2012). The teamwork quality approach focuses on the internal dynamics of the team.

When focusing on high-performing mature DevOps organizations, we assume the personal and cultural transformations have been completed on earlier DevOps transformation stages, as required by the methodology, and the quality of teamwork is a constant and can be put outside of the scope of this research.

2.5 Expertise composition research

In the pre-DevOps development, self-organizing agile teams are a well-researched and explained topic. Hoda and Noble attempted to find the best role composition for such teams, where by role they meant non-technical team roles such as coordinator, mentor, translator, champion, promoter and terminator (Hoda, Noble, & Marshall, 2013). These roles, however, assumed a fixed and separated professional roles in the team composition – such as developer, tester, project manager.
The overall field of self-organizing teams research stopped gaining traction as more organizations switched to the DevOps models and multi-skilled engineers – the need for such self-organization was discarded.

In 2013, Facebook was the first to achieve the superiorly higher rates of releases with a good performance and quality while having no dedicated QA department, no integrated quality engineers into the teams and sharing the QA responsibility between developers (Feitelson et al., 2013). Facebook engineers were instructed to write unit tests for all new code and expand the base of automated integration and acceptance tests. Developers were given a 6-week bootcamp that encouraged them to produce their first code change and commit it to production. The aim of this practice was to enable the culture in which mistakes are not punished and committing to production is an ordinary process.

The initial DevOps practices moved infrastructure operations into developers’ area of responsibility. Facebook went further and shared QA and other functions between developers. Maintaining CI/CD pipeline – a work that in some organizations is done by dedicated Build engineers, Releases, Security, Agile management – today, these functions in DevOps organizations are partially or fully executed by engineers (Pronschinske, 2019).

Studying the team composition as factor of performance was not in the scope of the existing theories until recently when DevOps and organizational transformations questioned a need for siloed Dev, QA, Ops and other functions of software development process and made it possible to combine them all in one team or one individual skillset (Wiedemann et al., 2019).

There is no common agreement neither in academia nor in the industry, whether to use QA, DevOps, Security or Release engineers or not in the development industry in the modern days, and how to balance the team composition – which creates an opportunity for this thesis’ research.

2.6 Theory summary

Overall, it was not possible to find one generic theory that covers optimization of performance in modern development using DevOps practices from the team composition and team culture perspective. The lack of such theory can be explained by the novelty of the DevOps approach, its continuous development and main focus on practical rather than theoretical aspects. Agile development research demonstrated a theoretical background behind why DevOps practices became successful in the modern days. The DevOps-methodology specific research is more scattered due to variance of results in applying DevOps to development practices and receiving non-homogenous results from performing the transformation. The practical evidences demonstrate positive organizational performance development for organizations that reached high levels of DevOps maturity – they release quickly, have established feedback loops, foster experimentation and collaboration culture, have visibility into product, infrastructure and customer needs.

Knowledge sharing and integration theory proposes measures of team collaboration and culture that are applicable to DevOps practices - such as individuality independence within the team context, team diversity, personality traits, skill levels and team roles.

Quality assurance theories provide a valuable historical insight into the quality definition and its metrics evolution. The most modern quality metrics were found to be bound to legacy development processes such as waterfall and tightly coupled with the requirements definitions and conformance to fixed customer expectations. It was shown that the key difference with DevOps is a continuous feedback loop between the organization and customers to capture and validate customer needs, roll
out new prototypes and release software that brings value to customers rather than satisfies a set of requirements. The new metric as a fusion of the classic quality approach and the novel DevOps and Agile quality methods needed to be proposed to capture the essence of quality in DevOps-practicing organizations.

Performance measures in development that exist in the literature originated from the traditional waterfall development research. They were transformed into the product management KPIs that are used to measure Agile team relative performance against the budget. These measures did not provide valuable insights from the DevOps performance point of view. The most applicable DevOps performance metrics were release frequency and change failure rate described by Forsgren which was taken as the basis for a new metric as a joint development performance indicator that unites independent metrics of release frequency and change quality.

Existing theories in the areas adjacent to this study provided a theoretical basis for determining the research problem, validating results, drawing conclusions from the data.

This research aims at making the following contribution to the development performance and DevOps transformation theory:

- Identify a need for a new measure of development performance in DevOps organizations and proposes it based on the existing Agile development theories and modern organizational performance research knowledge.
- Apply expertise and team composition theory to the newly defined performance metric.
- Verify a hypothesis that team composition and team culture influence development performance in DevOps organizations.
Chapter 3 Methodology

3.1 Methods overview

The choice of the research method is determined by the goal of the study – to find whether there exists a statistically significant relationship between development performance and factors such as expertise composition, team culture and various organizational factors. In order to achieve this goal, different research methodologies were reviewed.

Qualitative methods are intended to explore new problems, find promising research trends and can be used to provide the ground for quantitative methods (Ghauri & Grønhaug, 2005). In the case of this study, the research questions do not require additional qualitative research and therefore qualitative methods are not valid for this study.

Quantitative methods are based on collecting quantitative data and draw conclusions using statistical methods and generalization. The problem of the study is formulated in a quantitative way – we seek for a generalized relationship between performance and team composition taking into account side-effects of different development methodologies, company sizes and cultural differences in team work.

Quantitative methods have associated data collection methods that meet different needs and are suitable for different purposes.

User interviews is one of the reviewed methods for collecting research data. The interviews allow for getting insights from a single subject of the interview in a controlled setting with a possibility to clarify questions and receive the answers in desired form. This method is however time-consuming and requires significant effort from researcher and interviewee. The amount of data that can be collected using this method is limited to small samples (Shull, Singer, & Sjøberg, 2007). For the purpose of this study, it was required to collect diverse and representable data from organizations of different sizes, types, that work in various methodologies. Conducting interviews with team managers of software development teams would take lead to a rather small dataset limited to the 5-10 local companies.

Company survey is the method that allows for data collection with less effort required. A single survey is sent to all participants, and answers are received without researcher’s participation, on the contrary to the interviews. With significantly less effort, a study can gather much larger dataset. Among disadvantages of this methods is the potential ambiguity of the questions in a survey and inability to assist the recipient or ask for more details on particular topics. Another challenge with surveys is their low response rate – as low as 5% in software engineering surveys (Shull et al., 2007). The surveys are more suitable for large-scale data collection when research questions and questionnaires contain to ambiguity and provide well-articulated questions to the respondents. Also, the initial survey sample may be corrected based on the response rate from different subgroups to ensure the received answers are not unbalanced. For example, when sending a survey to the software industry, a researcher may realize that startup field is more responsive comparing to slower and more conservative enterprise sector and make amendments to the research design or the sample selection.

Case studies is a popular method in the business studies when it is required to study a particular phenomenon in its real-life context (Yin, 1992). The case study concentrates on the study subject and follows it closely over period of time. This method is divided into exploratory case studies that explore new theories based on studying a new phenomenon, and confirmatory case studies that
confirm existing theories. Case studies are suitable to studying long-lasting effects of a particular phenomenon over time or when the context plays a significant role in the experiment (Shull et al., 2007). Case studies are however vulnerable to selection bias and individual researcher perspective, they usually operate over a small sample size and require significant time and effort investments.

Controlled experiment is an efficient research method when experiment isolation and reproducibility is especially important, i.e. in the medical studies. The experiments are set up by a researcher and aimed at studying a response variable in a controlled setting where effects of other variables are minimized. Controlled experiments require significant effort from the researchers and are focused on the response variable in the research question. It is not applicable to situations when it is either impossible to remove impact of other variables or the experiment can’t recreate the phenomenon accurately.

Other research data collection methods include focus groups, brainstorming, and other methods.

For the purpose of this study, the company survey method was chosen due to its low effort and ability to cover wide range of participants regardless of their geographical location. The study relies on highly quantitative data from the release and quality functions that has the same meaning and no ambiguity between all participants.

3.2 Proposed measures

As discussed in the Theory chapter, the existing software team performance metrics are not directly suitable for reflecting the DevOps organizations performance due to their focus on traditional project delivery context rather than at the core of DevOps values – frequent releases and automation for operations and quality.

Burndown rates and velocity are not sufficient for answering the question whether a particular team is performing better comparatively to another team. Agile development items may be of different size, product development specifics may influence the burndown rates, and finally, the quality has no representation in these metrics. Lead time from idea to the customer is a novel measure which is very complex to measure – it requires multiple organizational departments to collaborate and trace the path of each work item. This measure also does not include a quality component.

Therefore, to better capture the organizational software development team performance, a new measure is proposed to operationalize the findings of the recent DevOps organizations research (Humble & Kim, 2018; Forsgren et al., 2019). As per Humble & Kim, and Forsgren, high-performing organizations release software as frequently and reliably as possible which in turn leads to better organizational performance and customer satisfaction. The proposed operationalization of this study is aimed at capturing both delivery frequency and the count of quality issues as continuous metric.

The new metrics is development performance score \( P \) which is estimated as the number of releases over a chosen time period \( NRel \) divided by the number of bugs, issues or problems reported for them \( NBugs \) by customers or discovered internally for these releases, the +1 is added to avoid division by zero in corner case when the releases had no bugs:

\[
P = \frac{NRel}{NBugs + 1}
\]

The intuition of this metrics is that the number of releases represents the team performance from the DevOps perspective which is built around smaller and more frequent releases (Brunnert et al.,
The classic release size parameters such as the number of lines of added code, number of releases features are deliberately not taken into account as they do not represent DevOps paradigm of minimizing the lead time of delivering the value to customers of the product. A large number of lines of code or added features may not convert into added value.

At the same time, DevOps culture not only encourages to release frequently but also focuses on keeping the quality at the acceptable level and building quality checks into the process. The motivation behind it is that with frequent but faulty releases, the customer satisfaction and trust level will quickly go down and will lead to the organization decline. Hence, the performance measure denominator is added as the sum of the number of issues in each release from the selected period. The measure of number of issues corresponds to known and widely used metrics in engineering quality literature.

The timescale for reporting the engineering performance in this research is set to a 3 month-period. This period is determined based on Agile and SCRUM sprint length which is usually set to 1-4 weeks with 2-3 weeks on average (Schwaber & Beedle, 2002). During three month-period, a team is capable of finishing 4-6 sprints that may have different focuses – quality, new features, continuous integration automation. Capturing multiple sprints outcome allows to smoothen discrepancies or irregularities associated with these types of activities.

The team composition in the existing literature is studied from various angles – such as professional composition, diversity, cross-functional composition. In the proposed DevOps setting where engineers by definition are multi-skilled, we look at the team professional composition within the following categories of five functions that to lesser or greater extent can be absorbed by the developers:

- Developer - represents a generic developer regardless of its title. For example, system developers, backend-, frontend- and full-stack developers are considered engineers. Seniority of a developer is a relative measure and is reflected in the job title.
- QA - a separate function solely focused on assessing quality of the products produced by Engineers. Examples – manual QA, automation QA.
- DevOps – support function that works with Infrastructure-as-Code, automated deployments, CI/CD, metrics and application feedback. Examples – DevOps engineer, Build and Integration engineer, Site reliability engineer.
- Security – product security function which ensures the product is delivered securely. Examples: security engineer, application security engineer, DevSecOps.
- Management function – the managerial functions directly related to the team such as product owner, product manager, agile coach. Scrum master role was also considered as part of the function.

Each of the functions is reported in the team composition with following metrics:

Number of engineers \((N_{Eng})\), number of senior software engineers \((N_{SEng})\), QA engineers \((N_{Qa})\), security engineers \((N_{Sec})\), DevOps engineers \((N_{Do})\), product owners, specialists and managers \((N_{Man})\) participating in designing, developing, delivering release to market. In this context, the number of senior software engineers is added to capture possible hierarchy level that may exist within the team.

The engineering composition and production rates are assessed on a per-team scale – for instance, if a DevOps engineer function is shared between 10 teams, the number of DevOps engineers in the team engineering composition would be 0.1. Same applies to managers, product owners, specialists.
etc. In some cases, scrum masters also take the developer role – this is reflected in the measurement according to the time spent in each role. For instance, if the scrum master activities take 20% of time, a team has 0.8 of developer and 0.2 of scrum master in the team.

The team composition and resource sharing are not the only factors that determine the productivity. In the modern DevOps environments, cross-team collaboration is highly encouraged. To capture the level of cross-team collaboration, the respondent-reported measure \( IndT \) is introduced. It is a relative coefficient of how integrated / independent a particular team is in the organizational context with possible values 1, 2, 3, 4 or 5 where \( IndT = 1 \) stands for a fully integrated team, where a team doesn’t produce any independent part of the product but rather a brings overall value and works with multiple other teams with no clear ownership boundaries. \( IndT = 5 \) is for fully independent teams, where each such team produces own product.

Another core value of DevOps culture is individual freedom of an engineer within the team – whether a developer can make technology-related decisions or is he or she supposed to ask for permission first. To capture the level of team self-sufficiency culture, the respondent-reported measure \( IndEmp \) is introduced. It is a relative coefficient of how independent and autonomous in their decisions are developers in the team context. The \( Ind \) coefficient has possible values 1, 2, 3, 4, 5. \( IndEmp = 1 \) stands for a fully dependent team members with all decision made by designated manager / leader and \( IndEmp = 5 \) is for fully engineers that can make the majority of day-to-day decisions on their own.

Both metrics \( IndEmp \) and \( IndT \) are derived from the Langfred’s 5-item autonomy scale used in surveys and interviews (Langfred, 2005).

The development framework and methodology provide additional insight into the team organization. It is captured as DevMethod and can take one of the values – SCRUM, Kanban, Spotify, SAFe + Scrum, SAFe + Kanban, LeSS, DAD. These values reflect the popular development methodologies as recognized by the literature (Alqudah & Razali, 2016). The survey also leaves space for customized implementation of any development methodology. The free text responses are also recorded and will later be assessed and generalized – i.e. “mix of Scrum and Kanban” can be classified as either Scrum or Kanban depending on the details provided, or a new class can be derived if more than one company shares a similar customized approach.

For the segmenting and descriptive statistics purposes, we record company size as the global headcount \( (NTot) \) and number of R&D employees \( (NRd) \).

Team size is recorded as: \( NT = NEng + NSEng + NQa + NSec + NDo + NMan \)

The product or development type \( (ProdType) \) is reported as one of the values: Web, Mobile, Embedded, Desktop, Service. Also, the open text option is provided, in the case a product cannot be categorized as any of these product types.

The team stability is an important research topic in the team composition research – the time in months from the last hire date is added to represent possible instability introduced by new team members and denoted as \( \text{LastHire} \).
3.3 Statistical modelling

Generalized linear model (Nelder & Wedderburn, 1972) was chosen as the initial point for modeling due to its flexibility when dealing with non-normal distribution of residuals, and also ability to transform it to one of its special cases – linear regression model or polynomial regression model.

Generalized linear model is the iterative weighted linear regression that “can be used to obtain maximum likelihood estimates of the parameters with observations distributed according to some exponential family and systematic effects that can be made linear by a suitable transformation” (Nelder & Wedderburn, 1972)

Another benefit of using a form of a linear or polynomial regression is the model interpretation simplicity. The resulting equation can be directly used to estimate and compare projected performance of teams with different expertise and culture composition.

3.3.4 Model variables

In the developed model, the DevOps performance score is a dependent continuous variable \( P \).

The following rates or fractions of each particular role to the total number of team players are recorded and used as independent variables.

- \( REng = NEng/NT \) – the rate of non-senior developers that are part of the team to the team size.
- \( RSEng = NSEng/NT \) - the rate of senior developers that are part of the team to the team size.

\( REng \) and \( RSEng \) were found to be highly correlated (Corr=0.8, see chapter 4 for details).

To avoid introducing multi-collinearity to the model (Johnston, Jones, & Manley, 2018), the two measurements were expressed via single variable \( RAEng \) that represented the rate of all engineers in the team, without capturing into the seniority levels.

- \( RAEng = (NEng+NSEng)/NT \) – the rate of all developers that are part of the team to the team size.
- \( RQa = NQa/NT \) - the rate of quality engineers that are part of the team to the team size.

Due to a limited sampling size and the highly zero inflated data for \( NSec, NDo \) and \( NMan \) (see Chapter 4 for details), a new generic measurement of all non-engineering functions was expressed as:

- \( ROth = (NSec + NDo + NMan)/NT \) - the rate of all non-developers that are part of the team to the team size, excluding QA.

Introduction of this variable allows to maintain higher number of degrees of freedom in data which is important in case a dataset has a small number of points (N=32). To illustrate the case for performing this transformation, the all non-composite variables model was tested and found to lack generalization capabilities (Appendix B, Table 5.14).

The other organizational factors were captured as following variables:

- \( IndT \) – relative level of cross-team collaboration, on the scale 1-5.
- \( IndEmp \) – relative level of individual autonomy within the team, on the scale 1-5.
The following organizational factors were captured in the survey but were excluded from the statistical model due to small sample size:

- DevMethod – development methodology, factor.
- LastHire – number in months since a list hire in the team, numeric variable.

3.3.5 Model equation

As the starting point for the model selection, a generic linear model that includes variables from the Chapter 3.3.4 is expressed as following:

\[ P = RAEng + RQA + ROTH + IndT + IndEmp \]

After performing linearity checks and verifying model assumption (see chapter 4), this model equation was rejected.

Multiple assumptions required to pass were not satisfied for using a linear model. To find the better model-data fit, higher-degree polynomials of the selected variables were introduced.

RQA was found to be highly correlated with RAEng (Corr=0.7) and was removed from the model. After the final model was found, RQA was added to it to assess its effects on the model parameters – as expected, it increased Variance Inflation but its value remained within the acceptable threshold.

Higher degree polynomial equations including the base variables were used, validated and the following model was chosen as the final model:

\[ P = RAEng + ROTH + ROTH^2 + ROTH^3 + ROTH^4 + IndT + IndEmp \]

To avoid multi-collinearity between four degrees of the same ROTH variable when using it in the model, orthogonal transformation was applied to ROTH (Kleinbaum, Kupper, Muller, & Nizam, 1988). All polynomial components ROTH were supplied to the model through R-language function “poly” that creates polynomials suitable for use in regression models, when applied with the parameter Raw=TRUE. Additionally, Variance Inflation Factor values were evaluated for all models to ensure the final model is not affected by multicollinearity. Skewness, kurtosis, heteroscedasticity were also checked for all models.

The main hypothesis of the model:

Developer, QA, other engineering roles composition, team or employee Independence do affect Engineering Performance.

Additional variables NTs, LastHire, DevMethod were tested separately after the base model is validated by adding them to the model and verifying model fit changes.

The proposed statistical model is set to answer the following research question:

Whether there exists a significant relationship between the development performance (expressed in terms of release frequency and defect counts) and such variables as team professional roles

---

composition, team culture and organizational factors of highly performant product-developing IT companies across the world?
3.4 Sample selection

The selection of sample for the research is limited to organizations that perceive themselves as high-performing and practice DevOps, release their own product software frequently and have completed or are at the latest stages of their DevOps and digital transformation journeys. This is done to avoid the research scope shift towards the traditional waterfall organizations that have completely different processes and development efficiency metrics.

The sample consists of 120 organizations across the world, the survey was sent to companies with head offices in Sweden, Finland, United States, Ukraine, Russia, Czech Republic, Netherlands, U.K., Germany, Australia.

The organizations were selected by the thesis author based on the information about their development process and DevOps maturity from the open sources. The qualifying organizations were determined based on available publications that describe current engineering practices and can be considered as mature DevOps organization or a control group (Humble & Kim, 2018). The sample selection included multi-national companies and aimed at capturing a wide variety of teams with their own development specifics that may influence software development performance. The resulting response rate was 30%, due to a small sample size no random re-sampling was performed.

The participating organizations varied in sizes and profiles – from startups to established enterprise leaders (see Chapter 4 for details). The design allows to draw descriptive statistics conclusions based on the company size and its stage of maturity.

The sample included Web, Embedded, Telecom, Mobile development team types.

Each organization was asked to submit an information on one most performant or highly performant team and its composition. There was no criterion given to the participants to determine whether a team is highly performant, they used their own judgement and subjective assessment of performance of a team comparing to other teams in the organization.

3.5. Survey design

The survey was designed according to the literature best practices such as six principles of survey design (B. A. Kitchenham & Pfleeger, 2002) to ensure a high response rate and motivation to respond, reproducibility, reliability, data validity, clarity of questions and their correspondence to the research goals.

Each organization was requested to fill in the survey for at least one team which is considered best performing. The results of multi-team surveys from the same responder would be averaged by the number of teams. For the example of the survey form, see Appendix A.

The survey recipients were defined as program and product managers, team managers, CTOs and senior technology management employees – these roles have access to the required data or can delegate the task to their reportees.

The questionnaire was proof-read by a professional engineering manager to ensure that the target audience understands and can answer the questions.

Privacy considerations were taken into account. No personal details or even company details were required as mandatory for submitting by respondents – participants were able to their company name, location or personal email for receiving a short research summary afterwards.
The survey was posted online and was made available to all participants via a short link. The link was sent to the target audience in companies over personal messages, company contact emails, published on a local developer community website (Reddit) and Twitter. For the latter case, a thorough vetting of replies was performed to ensure that the data submitted via the social media channels is realistic and reliable.
Chapter 4 Descriptive statistics and model reliability checks

Over the 3 months period, the dataset with 36 observations was collected which represented 30% response rate, 4 of the data entries were found ineligible (Figure 4.1) as they missed key values for releases or software defects, or did not pass the data sanity check.

![Figure 4.1 Distribution of eligible and non-eligible responses to survey](image)

Responses from Sweden, Finland, Ukraine, Russia, UK, Czech Republic, Philippines, Germany were received (Figure 4.2).

About 25% of all responses did not reference the country of their company origin or did not specify the company name which made impossible to determine the country of response.

![Figure 4.2 Location of responding team](image)
The half of the responses came from teams involved in the Web development. The overall product type distribution can be seen as per Figure 4.3.

![Figure 4.3 Distribution of product type among responders to the survey](image)

The responses were evenly distributed between companies of different size (Figure 4.4).

![Figure 4.4 Distribution of company sizes among responders to the survey](image)

The main development method is SCRUM and it is practiced by about a half of respondents, with Kanban being the second most popular with about a third of responses. Pure Agile, SAFe and Spotify are used less frequently (Figure 4.5).
Values of main variables has the following Mean, Median, Maximal and Minimal values (Table 4.1, Figure 4.6):

<table>
<thead>
<tr>
<th>Variable</th>
<th>Mean</th>
<th>Median</th>
<th>Max</th>
<th>Min</th>
</tr>
</thead>
<tbody>
<tr>
<td>NRel</td>
<td>31.94</td>
<td>11</td>
<td>200</td>
<td>1</td>
</tr>
<tr>
<td>NBugs</td>
<td>40.44</td>
<td>14</td>
<td>300</td>
<td>1</td>
</tr>
<tr>
<td>NEng</td>
<td>3.562</td>
<td>3</td>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td>NEng</td>
<td>2.188</td>
<td>2</td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td>NQa</td>
<td>0.9062</td>
<td>0</td>
<td>0</td>
<td>6</td>
</tr>
<tr>
<td>NDo</td>
<td>0.2</td>
<td>0.5219</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>NSec</td>
<td>0.1525</td>
<td>0.04</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>NMan</td>
<td>1.141</td>
<td>1</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>NTot</td>
<td>8.281</td>
<td>7</td>
<td>22</td>
<td>4</td>
</tr>
<tr>
<td>IndT</td>
<td>3.469</td>
<td>3</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>IndEmp</td>
<td>3.219</td>
<td>3</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>P</td>
<td>1.59</td>
<td>0.51</td>
<td>5.45</td>
<td>0.05</td>
</tr>
</tbody>
</table>

*Table 4.1 Descriptive statistics values for main variables*
Average Performance value is 1.59 which indicates higher number of releases than bugs discovered, with mean number of releases about 32 per 3 months which corresponds to releasing more than twice a week on average, with median value 11 releases per 3 months (releasing approximately every week). These numbers demonstrate that the target group of the survey is high-performing according to the definition defined in Chapter 3 of the thesis.

Median number of non-senior developers per team is 3 and senior developers - 2. Teams with no senior engineers also were reported.
Average number of QA engineers is less than 1 per team. This can be explained by the target group of this survey – organizations who are already mature in their DevOps practices, and tend to have less QA engineers comparing to the traditional waterfall development team numbers.

DevOps engineers are usually shared between then teams and do not belong to a team – with median number being 0.5 and average 0.2 (corresponds to sharing one DevOps resource among 2 or 5 teams correspondingly).

Security engineers are mostly absent from development – median value is 0.04 which means that one engineer is shared between 25 teams.

Managerial roles are quite common to be present in the teams with average 1 PM, Manager or Scrum Master being a part of a team. In some corner cases, this value is 0 (completely self-driven teams) or 3 – completely managed.

Total number of employees in an average team is 8 people, with median equal to 7. This shows that high performing development is usually done in small teams.

Team and employee independence are on average moderate and equal to 3 out of 5 for both variables. This indicates that independent teams and employees are well-represented in the organizations that consider themselves as high-performant ones. Surprisingly, there is no correlation (Pearson correlation coefficient = 0.06) between team and individual independence – high degree of team independence does not imply individual freedom and high degree of employee freedom does not make teams independent.
4.1 Exploring visible relations between response and independent variables

This section explores visible relationship between main response variable ($P$) and its components ($NRel$, $NBugs$) and important independent variables. This data provides understanding of data and cannot be used to draw statistically viable hypothesis about the data set.

Rate of engineers has visible influence on all three parameters and positively affects Performance and number of Releases but also leads to increased number of bugs in the products (Figure 4.7).

![Figure 4.7 Relation plots of rate of engineers to main response variables](image)

Exploratory plots of relation of QA rate to response variable demonstrate no visible influence on Performance. There exists a weak negative relationship with the number of bugs – more QA, slightly less bugs. Weak negative relationship with the number of releases – more QA leads to slightly less frequent releases (Figure 4.8).

![Figure 4.8 Relation plots of rate of QA to main response variables](image)

For other non-developer roles, as their number increases Performance decreases. There exists a weak relationship between increasing the number of other roles that leads to lesser number of bugs but also less frequent releases (Figure 4.9).
The highest average performance was reported for Web products, with Service and Mobile types being the contenders with almost twice lower values. The worst performers are Component and CI.

Kanban was reported as a development method that yields the best performance, with Scrum standing in the second position. The original Agile was the least performant method.

The best performing organizations are 1000-5000 followed up by 250-1000 and 51-250 in size. The least performing are 1-10 people organizations (Figure 4.10).
4.2 Checking Linear Model assumptions

For performing analysis, generalized linear model was chosen with response variable \( P \), independent variables \( \text{REng, RSEng, RQa, R0th, IndT, IndEmp} \).

The model assumptions (Neter, Kutner, Nachtsheim, & Wasserman, 1996) regarding the response and independent variables are tested to identify eligible variables and, if necessary, perform transformation to ensure variables can be used in the selected model family. The model type validation is required to make sure selected model fits the collected data.

4.2.1 Response Variable Distribution

Response variable \( P \) representing development team performance should follow or be close to the normal distribution. To check this assumption, the histogram and density of \( P \) are plotted (Figure 4.11):

![Distribution of P](image1)

![Density of P](image2)

*Figure 4.11 Distribution and density of the response variable*
As can be seen from the Figure 4.7, the distribution of $P$ does not follow the normal distribution and most values fall into the interval $[0,1)$ with some spike around $[4,5)$. This shape of data requires transforming before being used in GLM. To find the best response variable transformation, both square root transformation and log transformation are performed, and their response variable histograms produce the following distributions (Figure 4.12):

Log-transformation is not improving the response variable distribution and can be discarded.
As can be seen from the Figure 4.13, the resulting distribution better suits the normal distribution better than the original response variable. Bimodality and right left skewness can be observed and will be taken into account in the later steps of the fitted model analysis.

Boxplots of $P$ and $\sqrt{P}$ as shown on the Figure 4.14 indicate that variation of values of variable $P$ significantly decreased after the transformation:
4.2.2 Assessing non-collinearity between independent variables

Linear regression requires independent variables to have little or no multi-collinearity. To test the assumption, Pearson correlation coefficient heatmap is plotted for all independent variables (Figure 4.15).

The heatmap indicates that variables $REng$ and $RSEng$ are highly correlated (Pearson coefficient = -0.8) and cannot be used together in the model. The new variable is created: $RAEng = REng + RSEng$, it represents the rate of all software developers in a team.

To reflect the changes, a new heatmap that includes $RAEng$ is plotted (Figure 4.16). Significant correlation (Pearson coefficient = 0.74) between $RAEng$ and $RQa$ suggests that $RQa$ cannot be used in modeling and need to be removed from the model (Dormann et al., 2013).

Correlation between $RAEng$ and $ROth$ (Pearson coefficient = -0.49) is just below the 0.5 threshold and implies that $ROth$ can be used in the model.

Other correlations are insignificant and not exceeding 0.3.
Figure 4.15 Correlation heatmaps for the initial variable set

Figure 4.16 Correlation heatmaps for different independent variable sets
4.2.3 Linear relation between independent variables and response variable

Linearity plot (Figure 4.17) indicate higher degree polynomial relationship between response and RAEng and ROth variables. IndEmp follows linear relation.

The observed relationship makes it impossible to use linear model and requires exploring polynomial models of various degrees. Given a limited number of points in the dataset, a model needs to maintain relatively high number of degrees of freedom.

The modeling approach is to prioritize models with less variables and higher predictive power (higher R-squared) rather than with more variables (Colin Cameron & Windmeijer, 1997). In the case of a polynomial model, a model with a lower polynomial degree is preferred, given that it explains significant portion of variance. After the base model is found and tested, additional variables can be added one by one to it to test their significance.
4.2 Fitting and validating the polynomial model

In polynomial model fitting, multiple models of different degrees are built starting with the lowest polynomial degrees and adding additional degrees after testing model adjusted R-squared score. The best scoring model is chosen as the basis. The comparison table of all tested models with their R-squared:

<table>
<thead>
<tr>
<th>RAEng degree</th>
<th>ROther degree</th>
<th>IndT degree</th>
<th>IndEmp degree</th>
<th>Adj. R-squared</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>0.023</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>-0.011</td>
</tr>
<tr>
<td>1</td>
<td>4</td>
<td>1</td>
<td>1</td>
<td>0.3296</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>0.1274</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>0.118</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>1</td>
<td>1</td>
<td>0.3584</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>1</td>
<td>1</td>
<td>0.3302</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>1</td>
<td>1</td>
<td>0.3583</td>
</tr>
</tbody>
</table>

Table 4.2 Polynomial model R-squared comparison

The models are fitted using lm() function of R and orthogonal polynomials.

The best performing models are denoted as:

Model 1 (R^2 = 0.5375):

\[ \text{Sqrt}(P) = RAEng + ROther + ROther^2 + ROther^3 + ROther^4 + \text{IndT} + \text{IndEmp} \]

Model 2 (R^2 = 0.5951):

\[ \text{Sqrt}(P) = RAEng + RAEng^2 + ROther + ROther^2 + ROther^3 + ROther^4 + \text{IndT} + \text{IndEmp} \]

Model 3 (R^2 = 0.571):

\[ \text{Sqrt}(P) = RAEng + RAEng^2 + RAEng^3 + ROther + ROther^2 + ROther^3 + ROther^4 + \text{IndT} + \text{IndEmp} \]

Model 4 (R^2 = 0.5427):

\[ \text{Sqrt}(P) = RAEng + RAEng^2 + RAEng^3 + RAEng^4 + ROther + ROther^2 + ROther^3 + ROther^4 + \text{IndT} + \text{IndEmp} \]

To determine the best model, ANOVA analysis is performed on fitted model objects in R (Table 4.3):

<table>
<thead>
<tr>
<th>Model</th>
<th>RSS</th>
<th>Degrees of freedom</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>7.9517</td>
<td>24</td>
</tr>
<tr>
<td>2</td>
<td>7.2930</td>
<td>23</td>
</tr>
<tr>
<td>3</td>
<td>7.2834</td>
<td>22</td>
</tr>
<tr>
<td>4</td>
<td>6.660</td>
<td>21</td>
</tr>
</tbody>
</table>

Table 4.3 ANOVA of four best performing models

The resulting statistics shows that models 1 to 4 do not demonstrate statistically significant differences. Model 2 can be named best performing. It significantly decreases the value of RSS (Residual Sum of Squares) and maintains degrees of freedom. Higher-degree model do not show significant improvements in RSS. Model 1 maintains the highest number of degrees of freedom and
is comparable in predictive power to Model 2. Hence, these models are selected for further regression analysis and model fitting and are referenced here and further by their corresponding number.
4.3 Statistical modeling results

Model 1: \( \text{Sqrt}(P) = RAEng + ROth + ROth^2 + ROth^3 + ROth^4 + IndT + IndEmp \)

Model 2: \( \text{Sqrt}(P) = RAEng + RAEng^2 + ROth + ROth^2 + ROth^3 + ROth^4 + IndT + IndEmp \)

Model 1 and Model 2 are further compared based on RSE, F-statistics, adjusted R-squared values and VIF (variance inflation factor) for polynomial variables – see Appendix B for full model run results.

<table>
<thead>
<tr>
<th>Model</th>
<th>R-squared</th>
<th>F-stat</th>
<th>RSE</th>
<th>VIF (RAEng)</th>
<th>VIF (ROth)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model 1</td>
<td>0.3296</td>
<td>3.178</td>
<td>0.5756</td>
<td>1.3369</td>
<td>2.0651</td>
</tr>
<tr>
<td>Model 2</td>
<td>0.3584 (+0.028)</td>
<td>3.165 (+0.013)</td>
<td>0.5631 (-0.0125)</td>
<td>2.02589 (+0.689)</td>
<td>2.8780 (+0.8129)</td>
</tr>
</tbody>
</table>

Table 4.4 Model fit summary statistics for Model 1 and Model 2

Both models produce similar values on the main statistic metrics except for VIF. Variance inflation is higher with Model 2; therefore, Model 1 is chosen as final.

To find the best fit, potential outliers need to be analyzed and removed if they strongly contribute to the model fit or introduce a non-linearity to residuals distribution. The Q-Q residuals plot shows that datapoint 31 is breaking the linearity requirement for residuals (Figure 4.18).

Scale-Location plot (Figure 4.19) visualizes whether the model is homoscedastic, such that residuals are spread equally and randomly along fitted values. Scale-Location plot in this case shows that standardized residuals are affected by point 31, a possible outlier.
To ensure the model fit is not affected by the outlier, point 31 is removed from the dataset, and the model is fit again (Appendix B – fitting the Model 1 without point 31):

Competing model comparison (Table 4.5):

<table>
<thead>
<tr>
<th>Model</th>
<th>R-squared</th>
<th>F-stat</th>
<th>RSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model 1</td>
<td>0.3584</td>
<td>3.178</td>
<td>0.563</td>
</tr>
<tr>
<td>Model 1, without point 31</td>
<td>0.4647 (+0.1063)</td>
<td>4.721 (+1.543)</td>
<td>0.513 (-0.5)</td>
</tr>
</tbody>
</table>

*Table 4.5 Comparison of models with and without the outlier*

The new model significantly improved R-squared from 0.3584 to 0.4647, decreased RSE and improved F-statistics values. It also demonstrates linearity of Q-Q residuals plot (Figure 4.20), close to linear Scale-Location plot (Figure 4.21) and acceptable Residuals-Leverage values (Figure 4.22).
The resulting Scale-Location plot (Figure 4.21) shows that residuals are distributed nearly randomly across horizontal axis without strong irregularities or outliers.

Residuals vs Fitted plot (Figure 4.22) illustrates the difference between predicted values and the actual values of data. This plot is used to identify whether there is a good model-data fit.
Residuals vs Fitted plot shows outstanding data points that influence the model fit. The dotted lines correspond to Cook distance = 0.5 and 1.0. The Cook distance is calculated for each data point by removing it from the model and computing the regression (Chatterjee & Hadi, 1986). The result is compared to the original model. The Residuals vs Leverage plot has no outstanding data points. And the overall Leverage to Residuals relationship is close to linear.
Final checks on the overall model acceptance are performed using R function GVLMA (Pena & Slate, 2014) which evaluates model predictive power (Global Stat), Skewness of independent variables to response distribution, Kurtosis of independent variables to response distribution, continuity of response variable (Link Function parameter) and Heteroscedasticity – whether the variance between model residuals is constant. The function performs an evaluation of a null hypothesis and determines whether the given model satisfies the standards for these parameters, see Appendix B – GVLMA check for Model 1.

The results indicate that the model can be accepted as statistically significant as per all checks, it fits the distribution of data, no further outlier removal is performed. Potential removal of points 15 and 10 may have improved R-squared but would have also contributed to model overfitting to the given dataset. As the dataset size is very limited (31 points after removing the outlier), more generic model is preferred to more specific. This model explains 46.5% of variance in the response variable.

For the Model 1 after removing the outlier, the following coefficients and statistical significance values (P-values) were determined (Table 4.6):

<table>
<thead>
<tr>
<th>Coefficient</th>
<th>Estimate</th>
<th>P-value</th>
<th>Significant (p &lt; 0.05)?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intercept</td>
<td>1.053</td>
<td>0.21735</td>
<td>No</td>
</tr>
<tr>
<td>RAEng</td>
<td>-2.038</td>
<td>0.00655</td>
<td>Yes</td>
</tr>
<tr>
<td>ROth</td>
<td>46.04</td>
<td>0.00089</td>
<td>Yes</td>
</tr>
<tr>
<td>ROth^2</td>
<td>-460.8</td>
<td>0.00019</td>
<td>Yes</td>
</tr>
<tr>
<td>ROth^3</td>
<td>1439</td>
<td>0.00014</td>
<td>Yes</td>
</tr>
<tr>
<td>ROth^4</td>
<td>-1398</td>
<td>0.00017</td>
<td>Yes</td>
</tr>
<tr>
<td>IndEmp</td>
<td>0.4078</td>
<td>0.00160</td>
<td>Yes</td>
</tr>
<tr>
<td>IndT</td>
<td>-0.0592</td>
<td>0.46690</td>
<td>No</td>
</tr>
</tbody>
</table>

Model fit statistics

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>F-statistics</td>
<td>4.721</td>
</tr>
<tr>
<td>Adjusted R-squared</td>
<td>0.4647</td>
</tr>
<tr>
<td>RSE</td>
<td>0.513</td>
</tr>
</tbody>
</table>

Table 4.6 Polynomial regression Model 1 coefficients after removing the outlier

Independent variables $RAEng$, $ROth$ and $IndEmp$ are statistically significant ($p < 0.05$) and influence Performance ($P$) as response variable.

The final base model can be built using coefficients from the model output:

$$P = -2.038 \times RAEng + 46.04 \times ROth - 460.8 \times ROth^2 + 1439 \times ROth^3 - 1398 \times ROth^4 + 0.4078 \times IndEmp$$

4.3.1 Testing additional variables

Independent variable $LastHire$ is added to the model to test the significance of maintaining stable teams. The new model equation:

$$RAEng + ROth + ROth^2 + ROth^3 + ROth^4 + IndT + IndEmp + LastHire$$
The statistical summary indicates that LastHire has no statistically significant (p-value < 0.05) effect on Performance (Appendix B – Testing additional variables). R-squared of this model decreased despite adding an independent variable which indicates that LastHire does not statistically explain variance in Performance.

Independent variable NT is added to the model to introduce the significance of team size to the model. The new model equation:

\[ RAEng + ROTH + ROTH^2 + ROTH^3 + ROTH^4 + \text{IndT} + \text{IndEmp} + NT \]

The statistical summary indicates that NT has no statistically significant (p-value < 0.05) effect on Performance (Appendix B – Testing additional variables). R-squared of this model increased by 0.13 which is not significant given the addition of an independent variable. It shows that NT does not statistically explain variance in Performance.

Independent variable DevMethod is added to the model to test the significance of development methodology. The new model equation:

\[ RAEng + ROTH + ROTH^2 + ROTH^3 + ROTH^4 + \text{IndT} + \text{IndEmp} + \text{DevMethod} \]

The statistical summary table shows that DevMethod has no statistically significant (p-value <0.05) effect on Performance (Appendix B – Testing additional variables). R-squared of this model is lower than the original base model.

Testing RQa in the final model

After defining the base model is completed, it becomes possible to add RQa to assess its influence on the model fit parameters.

The new model equation:

\[ RAEng + RQa + ROTH + ROTH^2 + ROTH^3 + ROTH^4 + \text{IndT} + \text{IndEmp} \]

Adding the RQa term slightly the Adjusted R-squared measure comparing to the original model score, it performs worse in terms of F-statistics and slightly improves RSE value (Table 4.7 and Appendix B – Testing additional variables).

RQa is reported as not significant (p = 0.1033).

<table>
<thead>
<tr>
<th>Model</th>
<th>R-squared</th>
<th>F-stat</th>
<th>RSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model 1, without point 31</td>
<td>0.4647</td>
<td>4.721</td>
<td>0.513</td>
</tr>
<tr>
<td>Model 1, without point 31, with RQa</td>
<td>0.4802 (+0.155)</td>
<td>4.464 (-0.257)</td>
<td>0.505 (-0.08)</td>
</tr>
</tbody>
</table>

Table 4.7 Model comparison after adding RQa interaction

However, the new model doubles Variance Inflation Factor comparing to the original model - due to collinearity of RAEng and RQa (see Appendix B, Table 5.13). Also, Scale-Location, Residuals-vs-Fitted, Residuals-vs-Leverage indicate a worsened model fit after adding RQa (Figure 4.24).

<table>
<thead>
<tr>
<th>Coefficient</th>
<th>Estimate</th>
<th>P-value</th>
<th>Significant (p &lt; 0.05)?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intercept</td>
<td>2.464</td>
<td>0.0838</td>
<td>No</td>
</tr>
<tr>
<td>RAEng</td>
<td>-3.467</td>
<td>0.0134</td>
<td>Yes</td>
</tr>
<tr>
<td>RQa</td>
<td>-2.553</td>
<td>0.207913</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>Value</td>
<td>P-value</td>
<td>Result</td>
</tr>
<tr>
<td>---------</td>
<td>--------</td>
<td>----------</td>
<td>--------</td>
</tr>
<tr>
<td>$ROth$</td>
<td>48.07</td>
<td>0.0005</td>
<td>Yes</td>
</tr>
<tr>
<td>$ROth^2$</td>
<td>-472.3</td>
<td>0.000141</td>
<td>Yes</td>
</tr>
<tr>
<td>$ROth^3$</td>
<td>1449</td>
<td>0.000125</td>
<td>Yes</td>
</tr>
<tr>
<td>$ROth^4$</td>
<td>-1394</td>
<td>0.000164</td>
<td>Yes</td>
</tr>
<tr>
<td>$IndEmp$</td>
<td>0.3573</td>
<td>0.006513</td>
<td>Yes</td>
</tr>
<tr>
<td>$IndT$</td>
<td>-0.0473</td>
<td>0.558717</td>
<td>No</td>
</tr>
</tbody>
</table>

Table 4.8 Final model statistics after adding RQa interaction

This model indicates that adding QA to the development team does not have a statistically significant effect on performance.

The original model has a better generalization capability and is not strongly affected by multicollinearity – it will remain the final base model. The results of the final base model augmented with RQa will be used to conclude the role QA in development process.

4.4 Analysis summary

The statistical model rejected the null hypothesis and demonstrated that Rate of developers in the team ($RAEng$), Rate of non-developers ($ROth$) and employee independence ($IndEmp$) within the team context were proven to be statistically significant for maximizing development performance.

Rate of QA ($RQa$) engineers was found to have no effect on the performance when added to the final base model. Other factors that have no statistical significance in the given model are Team cross-functionality which can be seen as team independence ($IndT$), Team stability or how long time has passed since the last hire ($LastHire$), Development method ($DevMethod$), Team size ($NT$). All these variables may have indicated some relationship when analyzed empirically but high variance in data made it statistically insignificant.
Using real-world data for estimating team performance suggests that a team composed of only developers ($RAEng = 0.99$) is yielding 0.43 score of development performance ($\text{Sqrt}(P)$), while a team composed of 90% of developers and 10% is spread between shared roles is estimated to have $\text{Sqrt}(P) = 1.5$. The model performs well in corner cases, such as having no developers in the team – when $ROth = 1$, $P = -371$ – the team composed of non-developers only cannot function and produce output according to the model estimation.

The descriptive and exploratory analysis has shown that Web, Service and Mobile products demonstrate higher level of performance while Component and CI are lower. Kanban and SCRUM development methods demonstrated higher performance than Agile – while all these findings were not statistically significant in the model due to the high level of variance among the observations.

The organization size of 1000 to 5000 with 1-10 being the lowest performer. Similarly, this finding was only supported by descriptive analysis, modeling did not prove significance of the relationship.
Chapter 5 Conclusions

This research thesis explored effects of team composition and organizational culture on development performance in organization practicing DevOps and aiming at increasing release frequency and decreasing the number of defects per release. The novel measurement of development performance was proposed, and it is aimed at organization practicing DevOps principles alongside with modern Scrum, Kanban or Agile development methodologies. A statistical polynomial regression model was built, and statistical significance of various independent variables related to the modern development process was analyzed. The null hypothesis (rate of developers, non-developers and employee independence do not affect engineering performance) was rejected.

The main research question of the study: “Whether there exists a significant relationship between the development performance (expressed in terms of release frequency and defect counts) and such variables as team professional roles composition, team culture and organizational factors of highly performant product-developing IT companies across the world?”, was answered as following:

Development performance is affected by rate of developers, non-developers and individual autonomy within the team. Other variables do not significantly affect performance.

The developed model provides an equation that can take the real-world numbers as inputs and rates from existing teams and calculated their projected performance, it can also be used for composing new or modifying existing development teams.

As suggested by the model output, the optimal relation of developers to non-developers is about 9:1. Increasing or decreasing this proportion leads to a drop in the performance. The percentage of other engineering roles is relatively low and indicates that the roles such as DevOps, Security, Management can be shared among small development teams and play the role of architects or principal engineers in their field while outsourcing the routine DevOps, Security and Management tasks to the developers in the team.

The 9 to 1 rate comes close to the existing empirical data (Humble & Kim, 2018) that recommend so-called “shift-left” in the modern development frameworks (Lines & Ambler, 2015) which is based on utilizing developers that have multiple specializations and moving more outside functions into the team. However, it does not support the extreme scenarios such as Facebook’s all-in-one developers (Feitelson et al., 2013) in which developers are responsible for everything, from code to the release and operations. This research shows that some involvement of other roles is always required. Fully autonomous teams isolated from the business do not perform well, according to the model.

The optimal team independence score is 5 out 5. Employee independence plays significant role – the higher it is, the better performance teams demonstrate. This result falls in line with the empirical evidence of the recent DevOps surveys conducted by Kim and team culture research which indicates that higher employee autonomy leads to higher performance (Basaglia et al., 2010). The previous research proved this for self-reported efficiency and in the context of IT knowledge integration. This thesis expands the team autonomy research and shows that employee autonomy is statistically significant for developing product of good quality and releasing it frequently.

Due to the rate of quality engineers being highly correlated with the developer rate, it was not possible to reliably estimate the full magnitude of influence of QA engineers on the performance as independent variable. Nevertheless, after finding the final base model, the QA engineers rate
was added to it - and its effect has been found not significant. It is feasible to conclude that having or not having specialized QA engineers remains an open question and can be decided based on the product and quality needs, regulations and policies. This contradicts pre-Agile quality research theories which always assumed presence of QA engineers in the development process.

Degree of team integration into the larger development process was found non-significant. This finding shows that not only fully self-sufficient teams can perform well in DevOps settings but also fully integrated and dependent on inputs and outputs of other teams. The explanation to this can be seen in the sample selection – all organization perceive themselves as mature DevOps organizations. This fact implies transparent team boundaries, quick flow of product components between teams and lightweight processes.

5.1 Implications and contributions of the study

This study contributes to the research of development and organizational performance in the modern software organizations and provides an evidence on significant factors of development team composition and team culture affecting the development performance.

The research conducted in the project confirms and supports empiric findings industry leaders such as Facebook (Feitelson et al., 2013) and Flickr with regards to developer-centric team composition and enhanced performance. This thesis research goes one step further and finds the optimal rate of developers in a team - which challenges the Facebook’s assertion of having developer-only teams - this approach would not produce the optimal performance without shared supporting roles. Their findings, however, can be seen as ground-breaking for their time. Since then, there were no new publications related to Facebook development organization – it is presumable that the company adjusted the all-developers teams and given them access to shared functions, which would make it comply with the findings of this research. As the study suggests that the developer-centric teams with up to 90% of total team headcount being developers and shared DevOps, Security and Managerial roles maximize the number of releases while minimizing the number of associated defects. This contradicts earlier scientific publications on classic Scrum or Agile development organization that suggests keeping the specialists in the teams which was a standard practice for that time (Smits & Pshigoda, 2007). The ownership of the product and deliverables has shifted towards the team in early 2010-s and some publications started to recommend readjusting specialists knowledge, responsibilities and collaboration practices to keep the ownership of deliverables within the team and build a stronger ownership culture (Saddington, 2012). As DevOps concept introduces a full ownership of the developed component, it can be seen as the next step of ownership evolution – hence, efficient DevOps development requires transition to developer-centric teams that have skills and knowledge within the team. This shift eliminates specialist roles from the teams and the need for task handovers between roles that cause ownership dilution.

Contrary to the classic quality-related research, QA engineers, as found in the thesis research, do not directly influence development performance and quality in DevOps, their presence in the teams is found to be optional. At the same time, a closer inspection of the discoveries of the Agile Quality Assurance theory allows to see that the recent and modern publications suggest integrating Quality into development lifecycle as early as possible and testing small increments of development work (Bhasin, 2012). DevOps advocates for the same principles and promotes using the automation. The shift to not using specialized QAs is a part of the full ownership paradigm – for a developer, it is simpler and more reliable to write automated tests to their own code rather than explain and hand it over to a QA engineer. This is reflected in the latest State of DevOps 2019 research that shows
that all high and elite performers achieve high level of quality by using automated tests and automated unit tests written by developers (Forsgren et al., 2019).

High development performance score \((P)\) can be reached within different product types, company sizes and modern advanced Agile-based development methodologies. On the contrary to the expectation, this also applies to product types such as Embedded systems and Desktop applications that are known to have longer release cycles due to the nature of their deployment to physical devices on the customer side rather than Cloud-based deployment that are easier to perform and control. This finding supports Forsgren’s “State of DevOps” research discovery that states that any organization can become a high-performer by going through the DevOps transformation.

The study confirms the research findings of Edmondson on team and individual autonomy and fluid team boundaries (Edmondson & Nembhard, 2009) and places them into the modern context of teams that practice DevOps. A part of Edmondson’s findings on the team autonomy is contradicted – in the DevOps world it does not significantly affect performance. This means that both highly-isolated and integrated teams can achieve high DevOps performance scores. What matters is individual autonomy within the team – which is in line with Edmonson’s findings. The fluid team boundaries are part of DevOps knowledge sharing concept and can be seen as an essential part of building high-performant development organization. The thesis research confirms that teams of various size with multi-skilled developers that practice experimentation and encourage individual autonomy improve performance of all types and stages of development, including middleware or component development.

The classic development theory mentions team stability among significant factors for successful development outcome (Hoegl & Gemuenden, 2001). This can be explained by a low ownership culture where it requires a long time for a contributor to become efficient, and then adding or removing experienced members significantly affects the development performance. On the contrary to it, the thesis research found that in DevOps organizations, new members do not directly decrease team performance as long as the core team remains at least partially stable and there are team members that can share knowledge and enable new comers. The ownership culture allows to avoid irreplaceable contributors with unique domain knowledge – which follows the suggested principles of knowledge sharing and management in the development teams (Dingsøyr & Dybå, 2012). This leads to improved resiliency of high-performance development teams in which new members are able to become equal contributors within the shortest period of time, as compared to waterfall teams from the past that suffered significant performance degradation from team instability. In the modern development organizations that practice DevOps, new developers are expected to commit the code to production systems within their first week at the job (Sato, 2014).

Due to the lack of DevOps-specific measures of development productivity, a new measure of development performance proposed in the thesis can be adopted by the organizations for measuring their process efficiency and scientists for expanding DevOps performance research domain, independently from the conclusions or results of this research. This measure is in line with what Forsgren advocates for in the “State of DevOps” research but expresses it in quantitative domain – optimizing for release frequency which requires increased automation, which in turn would bring higher quality and lower number of bugs.

Overall, the main implication of the research is successful verifying and validating a set of assumptions for team composition and culture that will help organizations in their early and medium stages of DevOps transformation to build highly performing teams and minimize losses while reaching the high-performance cluster. These assumptions support empiric evidence from
industry practitioners and confirm or extend existing software development and team composition theories with the new angle of DevOps-specific performance and team structure. With the statistically verified and quantifiable tool such as the model equation, these organizations can improve their development performance, create highly performing teams and adjust the composition and culture of existing organizational units. This would lead to decreased financial losses while on the transformation journey and improved economy in the digitalized world.

5.2 Study limitations and future directions of research

This study pointed out the gaps in existing software development theory research that does not cover recent rapid development of DevOps practices and transformation of existing development frameworks.

This study main shortcomings are:

- Small sample size due to the need to answer non-trivial questions about team composition, number of releases and defects. Larger sample size will produce a model with better generalization capabilities. Building models for different development methodologies, product types of organizations sizes was not possible due to the lack of samples in each of the categories.
- This study operates with the rate of developers of a team in bulk without distinguishing the roles within the team.
- Focus of the study on only mature DevOps organizations. Most struggle with performance and quality is experienced by less mature organizations. However, results of this study are not directly applicable to their work practices.
- QA-engineers involvement was demonstrated as non-significant due to the high variance in the numbers. More nuanced survey may provide more insights into the actual contributions of QA engineers to the product quality.
- Team culture in the study is captured by only two parameters, it does not include diversity, communication and soft skills levels and other team dynamics metrics.

A potential research direction is how non-mature DevOps organizations and traditional companies can transform their development more quickly and efficiently to reach the high development performance. It can cover team composition, culture changes across various quality and performance metrics.

QA engineering in modern development organizations is another open research topic. This research suggests that quality engineers are not necessarily being a mandatory part of the development team anymore. At the same time, it does not distinguish between automation and manual QA activities, and the proportion of each of these functions distributed between developers in the team. Also, it does not provide insights into what type of quality activities allows teams composed of developers only reach acceptable level of quality. Therefore, exploring modern QA engineering of companies that are renowned for their product quality can bring interesting insights to industry and science.

Team dynamics research in modern highly diverse teams with multi-purpose developers opens new opportunities. The old development model assumed high levels of specializations while modern development requires developers to perform quite a broad set of functions, including internal and external communication, managerial, DevOps and security activities.
Within-team skill composition is another angle for the future research. This study can answer question how to build a well-rounded developers and which developers to hire into the developer-heavy team.

In the modern development, new paradigms and shifts are created every day. Due to the speed of change, their influence on the global development is still not fully understood. Will more frequent releases lead to more bugs? It is possible. But the lead time of fixes is also minimized. At the end, a customer gets the most value in the shortest time. With the adoption of modern development principles by banks, aircraft manufacturers, defense departments – will the price of an error be the same low as with a consumer Web development?

As every new and still developing concept, DevOps spawns the whole bunch of research questions, including ethical and philosophical. This study attempts to answer the question of optimizing performance through building multi-skilled teams with autonomous employees – which may lead to good or bad outcomes, depending on the context of application and its tolerance to defects.
References


Appendix A: Survey design

Research: How to compose software development teams for better performance? (~ 10 min to answer)

This form collects the data for a Master’s thesis research "How team composition affects software team delivery performance" at Blekinge Tekniska Högskola (Sweden), conducted by an MBA student Igor Andrushchenko. The study aims to find out how to compose teams for the best performance - measured as a delivery rate divided by quality defects rate. Is it better to train swiss-knife engineers that can do everything or keep more specialized roles - and in which proportion?

This survey is confidential and anonymous by default - the research won't mention which practices belong to which organization. The overall results and conclusions will be shared with participants who left a contact information, for a potential benefit of their organizations and as a token of appreciation for filling out this form.

Responder profile: team leader / team manager / release manager / delivery manager / agile leader

Challenging questions: at one point, you will be asked to enter the number of releases over a 3 months period and the number of issues reported for these releases - digging up this information may require some effort.

You can always reach the researcher using contact details below if you have any questions.

Contact details:
Email: igor.segodnya@gmail.com
LinkedIn: https://www.linkedin.com/in/igor-andrushchenko/

Disclaimer: Igor Andrushchenko works full-time as Director of Engineering Security at Snow Software. This research is conducted independently from the organization, is not sponsored nor supported by it. The collected raw data will not be shared with Snow Software.
About the organization

This section focuses on the global organizational profile.

Organization name (will remain confidential)
Your answer

Total number of employees in your organization (including all subsidiaries) *

- [ ] 1-10
- [ ] 11-50
- [ ] 51-250
- [ ] 250-1000
- [ ] 1000-5000
- [ ] 5000+

... of which working for R&D? *

- [ ] 1-10
- [ ] 11-50
- [ ] 51-250
- [ ] 250-1000
- [ ] 1000-5000
- [ ] 5000+

Responder contact (for results report)
Your answer
Team profile

To answer, you should pick one of the best-performing product teams that releases frequently.

Then, pick a 3-month period which reflects the stable and steady functioning of the team (there were no big changes in the organization or team composition).

Product type that team works on *

- Web
- Desktop
- Mobile
- Embedded
- Service
- Other: ____________________________

Development methodology / framework used by the team *

- Scrum
- Kanban
- Agile
- Spotify
- SAFe + Scrum
- SAFe + Kanban
- LeSS
- DAD
- Other: ____________________________

Methodology specifics (if any)

Your answer

Total number of employees in the team? *

Your answer
Time in months since the last hire in the team? *

Your answer

How independent is the team in the organizational context? (0 = team does only internal deliveries, has vital dependencies on other teams. 5 = team develops a product completely on its own) *

Fully integrated

1 2 3 4 5

Fully independent

How independent is an individual in the team context? (can an engineer make an important product decision or does he/she have to ask for a permission?) *

Decisions are made by lead/manager

1 2 3 4 5

Decisions are made by developers (self-driven)

Team performance

To answer, pick one of the best-performing product teams that releases frequently.

Then, pick a 3-month period which reflects the stable and steady functioning of the team (there were no big changes in the organization or team composition).

Estimated number of releases over chosen 3 months-period (including minor and patches) *

Your answer

Estimated number of bugs / issues reported or discovered over this period *

Your answer
Team composition

Number of software developers in the team *
Your answer

Number of senior or lead software developers in the team *
Your answer

Number of QA engineers in the team (if function is shared with other teams, use fraction - i.e 0.1 for 1 QA / 10 teams) *
Your answer

Number of DevOps / SRE / Build engineers in the team (if function is shared with other teams, use fraction - i.e 0.1 for 1 DevOps / 10 teams) *
Your answer

Number of Security engineers in the team (if function is shared with other teams, use fraction - i.e 0.1 for 1 Security / 10 teams) *
Your answer

Number of managerial roles in the team (if function is shared with other teams, use fraction - i.e 0.1 for 1 manager/PM /PO/Agile coach who works with 10 teams) *
Your answer

Notes on the team composition (if something isn't covered by the questions above)
Your answer
Appendix B: Statistical modeling details

ANOVA of Models 1-4

Analysis of Variance Table

<table>
<thead>
<tr>
<th>Res.Df</th>
<th>RSS</th>
<th>Df</th>
<th>Sum of Sq</th>
<th>F</th>
<th>Pr(&gt;F)</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>7.95</td>
<td>1</td>
<td>0.65878</td>
<td>2.0773</td>
<td>0.1643</td>
</tr>
<tr>
<td>2</td>
<td>7.29</td>
<td>1</td>
<td>0.00958</td>
<td>0.0302</td>
<td>0.8637</td>
</tr>
<tr>
<td>3</td>
<td>7.28</td>
<td>1</td>
<td>0.00958</td>
<td>0.0302</td>
<td>0.8637</td>
</tr>
<tr>
<td>4</td>
<td>6.66</td>
<td>1</td>
<td>0.62340</td>
<td>1.9657</td>
<td>0.1755</td>
</tr>
</tbody>
</table>

Table 5.1 ANOVA of four best performing models

Comparing polynomial models

Model 1: $\text{Sqrt}(P) = RAEng + ROTH + ROTH^2 + ROTH^3 + ROTH^4 + \text{IndT} + \text{IndEmp}$

Model 2: $\text{Sqrt}(P) = RAEng + RAEng^2 + ROTH + ROTH^2 + ROTH^3 + ROTH^4 + \text{IndT} + \text{IndEmp}$

The polynomial regression model fit summary for Model 1:

<table>
<thead>
<tr>
<th>Residuals:</th>
<th>Min</th>
<th>1Q</th>
<th>Median</th>
<th>3Q</th>
<th>Max</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>-1.28858</td>
<td>-0.23658</td>
<td>-0.00611</td>
<td>0.27420</td>
<td>1.15933</td>
</tr>
</tbody>
</table>

Coefficients:

| Estimate | Std. Error | t value | Pr(>|t|) |
|----------|------------|---------|---------|
| (Intercept) | 1.102e+00 | 9.310e-01 | 1.184 | 0.24804 |
| poly(RAEng, 1, raw = TRUE) | -1.737e+00 | 7.546e-01 | -2.301 | 0.03036 * |
| poly(ROth, 4, raw = TRUE) | 3.981e+01 | 1.330e+01 | 2.994 | 0.00629 ** |
| poly(ROth, 4, raw = TRUE) | -4.112e+02 | 1.147e+02 | -3.584 | 0.00150 ** |
| poly(ROth, 4, raw = TRUE) | 1.307e+03 | 3.507e+02 | 3.727 | 0.00105 ** |
| poly(ROth, 4, raw = TRUE) | -1.286e+03 | 3.470e+02 | -3.705 | 0.00111 ** |
| IndEmp | 3.437e-01 | 1.251e-01 | 2.747 | 0.01121 * |
| IndT | -5.217e-02 | 8.971e-02 | -0.582 | 0.56627 |

---

Signif. codes: 0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.5756 on 24 degrees of freedom
Multiple R-squared: 0.481, Adjusted R-squared: 0.3296
F-statistic: 3.178 on 7 and 24 DF, p-value: 0.01588

Table 5.2 Polynomial regression Model 1 statistical summary

The model fit summary for Model 2:

<table>
<thead>
<tr>
<th>Residuals:</th>
<th>Min</th>
<th>1Q</th>
<th>Median</th>
<th>3Q</th>
<th>Max</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>-1.16451</td>
<td>-0.25507</td>
<td>-0.01948</td>
<td>0.31312</td>
<td>1.07085</td>
</tr>
</tbody>
</table>

Coefficients:

| Estimate | Std. Error | t value | Pr(>|t|) |
|----------|------------|---------|---------|
| (Intercept) | 3.822e+00 | 2.095e+00 | 1.824 | 0.08114 . |
| poly(RAEng, 2, raw = TRUE) | -1.086e+01 | 6.375e+00 | -1.704 | 0.10184 |
| poly(RAEng, 2, raw = TRUE) | 6.515e+00 | 4.520e+00 | 1.441 | 0.16295 |
| poly(ROth, 4, raw = TRUE) | 3.833e+01 | 1.305e+01 | 2.938 | 0.00739 ** |
| poly(ROth, 4, raw = TRUE) | -3.745e+02 | 1.151e+02 | -3.253 | 0.00350 ** |
| poly(ROth, 4, raw = TRUE) | 1.164e+03 | 3.572e+02 | 3.258 | 0.00346 ** |
| poly(ROth, 4, raw = TRUE) | -1.128e+03 | 3.568e+02 | -3.161 | 0.00437 ** |
| IndEmp | 3.858e-01 | 1.258e-01 | 3.066 | 0.00547 ** |
| IndT | -6.263e-02 | 8.806e-02 | -0.711 | 0.48411 |

---

Signif. codes: 0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ‘ 1
Models produce similar F and P-statistics values, RSE and R-squared. When fitting polynomial models, it is important to verify the Variance Inflation Factor values and ensure that higher degree polynomials do not contribute to model's multicollinearity (Craney & Surles, 2002).

Model 1 performs as following:

\begin{verbatim}
poly(RAEng, 1, raw = TRUE)  1.336967  1        1.156273
poly(ROth, 4, raw = TRUE)   2.065125  4        1.094884
IndEmp                     1.486746  1        1.219322
IndT                       1.165115  1        1.079405
\end{verbatim}

Model 2 performs as following:

\begin{verbatim}
poly(RAEng, 2, raw = TRUE)  2.025895  2        1.193038
poly(ROth, 4, raw = TRUE)   2.878065  4        1.141268
IndEmp                     1.571137  1        1.253450
IndT                       1.173073  1        1.083085
\end{verbatim}

Model 2 VIF value for ROth component is larger than Model 1 (2.878 versus 2.065). Given that the data set is small, a conservative VIF threshold = 2.5 is chosen (Craney & Surles, 2002) to eliminate the model with higher multicollinearity bias.

Fitting Model 1 without point 31

The model run in R produces the following table of results:

\begin{verbatim}
Residuals:
Min -0.98523 -0.24802  0.01001  0.31854  1.03376
1Q   0.30812  0.48798  0.58406  0.58406  1.03376
Median 0.30812  0.48798  0.58406  0.58406  1.03376
3Q   0.52751  0.71240  0.90081  0.90081  1.03376
Max  1.03376  1.03376  1.03376  1.03376  1.03376

Coefficients:
(Intercept)                 1.053e+00  8.299e-01   1.268 0.217353
poly(RAEng, 1, raw = TRUE) -2.038e+00  6.818e-01  -2.989 0.006552 **
poly(ROth, 4, raw = TRUE) 1  4.604e+01  1.207e+01   3.813 0.000894 ***
poly(ROth, 4, raw = TRUE) 2 -4.608e+02  1.039e+02  -4.435 0.000190 ***
poly(ROth, 4, raw = TRUE) 3  1.439e+03  3.163e+02   4.548 0.000144 ***
poly(ROth, 4, raw = TRUE) 4 -1.398e+03  3.121e+02  -4.478 0.000171 ***
IndEmp                      4.078e-01  1.140e-01   3.576 0.001600 **
IndT                       -5.921e-02  7.999e-02  -0.740 0.466690
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.513 on 23 degrees of freedom
Multiple R-squared: 0.5896,  Adjusted R-squared: 0.4647
F-statistic: 4.721 on 7 and 23 DF,  p-value: 0.002096
\end{verbatim}
GVLMA check for Model 1

GVLMA function run produces the following output:

<table>
<thead>
<tr>
<th></th>
<th>Value</th>
<th>p-value</th>
<th>Decision</th>
</tr>
</thead>
<tbody>
<tr>
<td>Global Stat</td>
<td>4.352471</td>
<td>0.36040</td>
<td>Assumptions acceptable.</td>
</tr>
<tr>
<td>Skewness</td>
<td>0.008084</td>
<td>0.92836</td>
<td>Assumptions acceptable.</td>
</tr>
<tr>
<td>Kurtosis</td>
<td>0.008940</td>
<td>0.92467</td>
<td>Assumptions acceptable.</td>
</tr>
<tr>
<td>Link Function</td>
<td>3.128766</td>
<td>0.07692</td>
<td>Assumptions acceptable.</td>
</tr>
<tr>
<td>Heteroscedasticity</td>
<td>1.206682</td>
<td>0.27199</td>
<td>Assumptions acceptable.</td>
</tr>
</tbody>
</table>

Table 5.7 Assessment of general statistical acceptance parameters for Model 1

Testing additional variables with Model 1

Adding LastHire to the equation:

| Coefficients:     | Estimate | Std. Error | t value | Pr(>|t|) |
|-------------------|----------|------------|---------|---------|
| LastHire          | 4.023e-03 | 3.117e-02  | 0.129   | 0.898497|

Residual standard error: 0.5243 on 22 degrees of freedom
Multiple R-squared: 0.5899, Adjusted R-squared: 0.4408
F-statistic: 3.956 on 8 and 22 DF, p-value: 0.004915

Table 5.8 Statistics summary of Model 1 with added LastHire variable

Adding NT to the equation:

| Coefficients:     | Estimate | Std. Error | t value | Pr(>|t|) |
|-------------------|----------|------------|---------|---------|
| NT                | 4.216e-02 | 3.340e-02  | 1.262   | 0.220015|

Residual standard error: 0.5065 on 22 degrees of freedom
Multiple R-squared: 0.6174, Adjusted R-squared: 0.4782
F-statistic: 4.437 on 8 and 22 DF, p-value: 0.002581

Table 5.9 Statistics summary of Model 1 with added NT variable

Adding DevMethod to the equation:

| Coefficients:     | Estimate | Std. Error | t value | Pr(>|t|) |
|-------------------|----------|------------|---------|---------|
| DevMethodKanban    | 7.798e-01 | 5.955e-01  | 1.310   | 0.205966|
| DevMethodSAFE + Scrum | 5.010e-01 | 5.944e-01  | 0.843   | 0.409812|
| DevMethodScrum     | 3.558e-01 | 4.680e-01  | 0.760   | 0.456402|
| DevMethodSpotify   | 1.075e+00 | 7.946e-01  | 1.353   | 0.191907|

Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.5232 on 19 degrees of freedom
Multiple R-squared: 0.6474, Adjusted R-squared: 0.4432
F-statistic: 3.171 on 11 and 19 DF, p-value: 0.01327

Table 5.10 Statistics summary of Model 1 with added DevMethod variable
Final model - adding RQa to the equation

<table>
<thead>
<tr>
<th>Residuals:</th>
<th>Min</th>
<th>1Q</th>
<th>Median</th>
<th>3Q</th>
<th>Max</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>-0.89864</td>
<td>-0.28564</td>
<td>0.01284</td>
<td>0.28946</td>
<td>0.98249</td>
</tr>
</tbody>
</table>

Coefficients:

|                  | Estimate | Std. Error | t value | Pr(>|t|) |
|------------------|----------|------------|---------|----------|
| (Intercept)       | 2.464e+00 | 1.361e+00  | 1.811   | 0.083896 |
| poly(RAEng, 1, raw = TRUE) | -3.467e+00 | 1.290e+00  | -2.688  | 0.013444 * |
| poly(ROth, 4, raw = TRUE)1 | 4.807e+01  | 1.200e+01  | 4.005   | 0.000141 *** |
| poly(ROth, 4, raw = TRUE)2 | -4.723e+02 | 1.028e+02  | -4.595  | 0.000125 *** |
| poly(ROth, 4, raw = TRUE)3 | 1.449e+03  | 3.118e+02  | 4.646   | 0.000125 *** |
| poly(ROth, 4, raw = TRUE)4 | -1.394e+03 | 3.075e+02  | -4.534  | 0.000164 *** |
| IndEmp            | 3.573e-01  | 1.189e-01  | 3.005   | 0.006513 ** |
| IndT              | -4.713e-02 | 7.938e-02  | -0.594  | 0.558717 |
| RQa               | -2.553e+00 | 1.968e+00  | -1.297  | 0.207913 |

Signif. codes: 0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.5055 on 22 degrees of freedom
Multiple R-squared: 0.6188, Adjusted R-squared: 0.4802
F-statistic: 4.464 on 8 and 22 DF, p-value: 0.002491

Table 5.11 Statistics summary of Model 1 with added RQa

<table>
<thead>
<tr>
<th></th>
<th>Value</th>
<th>p-value</th>
<th>Decision</th>
</tr>
</thead>
<tbody>
<tr>
<td>Global Stat</td>
<td>3.99233</td>
<td>0.40704</td>
<td>Assumptions acceptable.</td>
</tr>
<tr>
<td>Skewness</td>
<td>0.05823</td>
<td>0.80932</td>
<td>Assumptions acceptable.</td>
</tr>
<tr>
<td>Kurtosis</td>
<td>0.15799</td>
<td>0.69101</td>
<td>Assumptions acceptable.</td>
</tr>
<tr>
<td>Link Function</td>
<td>2.80327</td>
<td>0.09407</td>
<td>Assumptions acceptable.</td>
</tr>
<tr>
<td>Heteroscedasticity</td>
<td>0.97284</td>
<td>0.32397</td>
<td>Assumptions acceptable.</td>
</tr>
</tbody>
</table>

Table 5.12 Assessment of statistical acceptance parameters for Model with RQa

<table>
<thead>
<tr>
<th></th>
<th>GVIF</th>
<th>Df</th>
<th>GVIF(^{(1/(2*Df))})</th>
</tr>
</thead>
<tbody>
<tr>
<td>poly(RAEg, 1, raw = TRUE)</td>
<td>5.031159</td>
<td>1</td>
<td>2.243025</td>
</tr>
<tr>
<td>poly(ROth, 4, raw = TRUE)</td>
<td>4.592109</td>
<td>4</td>
<td>1.209906</td>
</tr>
<tr>
<td>IndEmp</td>
<td>1.705946</td>
<td>1</td>
<td>1.306119</td>
</tr>
<tr>
<td>IndT</td>
<td>1.177029</td>
<td>1</td>
<td>1.084910</td>
</tr>
<tr>
<td>RQa</td>
<td>4.936137</td>
<td>1</td>
<td>2.221742</td>
</tr>
</tbody>
</table>

Table 5.13 VIF calculation for the final Model with RQa

This model surpasses a conservative VIF threshold = 2.5 (Craney & Surles, 2002).
Model with only original variables without composite measurements

Here, $R_{\text{Sec}} = \frac{\text{NSec}}{\text{NT}}$; $R_{\text{Man}} = \frac{\text{NMan}}{\text{NT}}$; $R_{\text{Do}} = \frac{\text{NDo}}{\text{NT}}$

| Coefficients | Estimate | Std. Error | t value | Pr(>|t|) |
|--------------|----------|------------|---------|----------|
| (Intercept)  | 3.9794   | 1.8290     | 2.176   | 0.0401 * |
| REng         | -2.8184  | 1.6844     | -1.673  | 0.1078   |
| RSEng        | -3.8772  | 1.8479     | -2.098  | 0.0471 * |
| RQa          | -2.8745  | 2.3970     | -1.199  | 0.2427   |
| RMan         | -2.4785  | 1.8717     | -1.324  | 0.1985   |
| RSec         | -0.8010  | 4.3160     | -0.186  | 0.8544   |
| RDo          | -0.4283  | 3.0201     | -0.142  | 0.8884   |
| IndT         | -0.1079  | 0.1075     | -1.004  | 0.3261   |
| IndEmp       | 0.1385   | 0.1560     | 0.888   | 0.3839   |

---

Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.6936 on 23 degrees of freedom
Multiple R-squared: 0.2777, Adjusted R-squared: 0.0265
F-statistic: 1.105 on 8 and 23 DF, p-value: 0.3953

Table 5.14 Statistics of the non-composite variables model

This model explains 2% of the variation in data, has low F-statistics values and cannot be accepted as it fails basic linear model checks.