A Taxonomy of Software Delivery Performance Profiles: Investigating the Effects of DevOps Practices

Nicole Forsgren
GitHub, nicolefv@gmail.com

Marcus A. Rothenberger
University of Nevada Las Vegas, marcus.rothenberger@unlv.edu

Jez Humble
Google LLC, humble@google.com

Jason B. Thatcher
Temple University, jason.thatcher@temple.edu

Dustin Smith
Google LLC, smithdc@google.com

A Taxonomy of Software Delivery Performance Profiles: Investigating the Effects of DevOps Practices[1]

Emergent Research Forum (ERF)

Nicole Forsgren
GitHub
nicolefv@gmail.com

Marcus A. Rothenberger
University of Nevada Las Vegas
marcus.rothenberger@unlv.edu

Jez Humble
Google LLC
humble@google.com

Jason B. Thatcher
Temple University
jason.thatcher@temple.edu

Dustin Smith
Google LLC
smithdc@google.com

Abstract

This research develops a taxonomy of Software Delivery Performance Profiles for DevOps development settings. We base the underlying Software Delivery Performance measure on the application of the Economic Order Quantity (EOQ) model to software development. Consistent with the objectives of both, development and operations departments, the measure includes attributes for throughput (release frequency and lead-time to delivery) and for stability (mean time to restore). Hierarchical cluster analysis on a global sample of 7,522 DevOps professionals results in three distinct Software Delivery Performance Profiles; in addition, the results indicate that the measures for throughput and stability move in tandem. Further analysis will show how the use of individual DevOps practices impacts Performance Profiles of development settings. When completed, the study will support the utility of DevOps and the effectiveness of individual DevOps practices.

Keywords

DevOps, Economic Order Quantity, cluster analysis, software delivery performance, transaction cost

Introduction

The concept of DevOps has been developed as an extension to the agile paradigm for IT service management (Debois 2011), which focused on developing new processes for the continuous deployment of rapidly changing software. By combining software development and IT Operations, DevOps implements a combination of cultural shifts and technology-enabled practices (Wiedemann et al. 2019) with the goal of achieving higher levels of both, throughput and stability, even in an environment of high uncertainty (Humble and Molesky 2011). However, little academic research has emerged that has examined the implications of individual DevOps practices on Software Delivery Performance (e.g., Leite et al. 2019). This study will work towards alleviating this shortcoming by using the Economic Order Quantity (EOQ) model as a theoretical framework to develop a taxonomy of Software Delivery Performance Profiles of DevOps

[1] An early version of this research stream appeared in the 2016 Proceedings of the Western Decision Sciences Institute Annual Meeting (Forsgren and Humble 2016).

development settings, and to research the effects of DevOps practices on the Software Delivery Performance Profiles. Hereby, the study will address the following research questions:

RQ1: Which Software Delivery Performance Profiles exist in development settings employing DevOps?

RQ2: How does the use of individual DevOps practices influence Software Delivery Performance?

This Emergent Research Forum (ERF) paper covers the development of the taxonomy that results from addressing RQ1. First, we draw on EOQ to create our Software Delivery Performance construct. Then we discuss the data collection and develop the taxonomy of Software Delivery Performance Profiles in DevOps settings using cluster analysis. The completed research will further address RQ2 by relating individual DevOps Practices to the Software Delivery Performance Profiles.

Economic Order Quantity

The Economic Order Quantity (EOQ) model was a precursor to Lean Manufacturing (e.g., Karlsson and Ahlstrom 1996; Kilpatrick 2003; Tang and Musa 2011) and was derived by Ford W. Harris (1913). It models the economic trade-offs between batch size and transaction cost, as well as holding cost. EOQ was created in the context of manufacturing, so it is important to apply its variables to software development. In this context, a *batch* is a system change that is modelled in terms of one or more version control commits, which describe the changes that will be made to one or more IT services or the infrastructure on which they run (Forsgren and Humble 2016). We define the *holding cost* as the opportunity cost of not completing and deploying a change, measured as the cost of delay (Reinertsen 2009). In addition, we define the *transaction cost* as the cost of completing and deploying a change. Accordingly, the sum of the transaction cost and the holding cost is the *total cost*. Consistent with EOQ, we assume that organizations behave rationally by choosing a batch size that minimizes *total cost* (Forsgren and Humble 2016).

Software Delivery Performance in the Context of DevOps

Software Delivery Performance in the context of DevOps must reflect what matters to development, as well as operations departments. Development can be evaluated in terms of code delivery *throughput*, while operations prioritizes the *stability* of infrastructure and software (Forsgren and Humble 2016; Humble and Molesky 2011). Therefore, following our previous section's discussion on EOQ, we use a performance measure that captures both of these dimensions.

Throughput Attributes

In order to develop the throughput measures, we view the software development process as a process similar to that of a manufacturing plant (Humble et al. 2006). We define batch size as the size of a system change (Forsgren and Humble 2016). However, it is difficult to objectively measure the size of a change in software development, which is a major problem when applying manufacturing principles to IT (Reinertsen 2009; Wang et al. 2012). Therefore, we use release frequency as a proxy for batch size (Forsgren and Humble 2016). In fact, in just-in-time manufacturing, release frequency (production rate) approaches infinity as batch size approaches zero (Khan and Sarker 2002), thus, release frequency is a suitable proxy measure. In software development, it captures how often code is deployed or released to the production environment. This measure can be easily collected, as it is highly visible and frequently used as a key performance indicator (KPI) for software delivery teams (Bird 2012; Radigan 2015). The other throughput measure is lead-time for delivery; in software development, this is typically called lead-time for changes. We define this measure as the time required for a change to software or infrastructure from its commitment to the central version control repository to running in production (including integration, testing, and quality assurance) (Forsgren and Humble 2016; Rother and Shook 2003). Lead time for changes is easy to measure and comparable across development settings (e.g., Khomh et al. 2012; Radigan 2015).

Stability Attribute

The second type of our performance measure is stability. We capture this by obtaining the time required to restore the system when the system is impaired or unavailable; this is the lead time of "emergency changes". Short times to restore indicate high levels of software stability in an organization. Therefore, we use mean time to restore (MTTR) to measure stability (Forsgren and Humble 2016); MTTR is the time required to restore software after an incident occurs that makes the system unavailable or impairs quality of service (Kullstam 1981). This measure is an industry standard and reflects the transaction cost of emergency changes well (Allspaw 2010).

Data Collection

Data from DevOps professionals were collected using a web-based survey. We have posted invitations to participate in this study in different outlets, including DevOps messaging boards and Twitter. Participants were also invited to refer colleagues who worked in DevOps to participate in the study. They were asked to provide data on their development settings that captured the attributes of the Software Delivery Performance measure discussed in the previous section. The survey was available over a four-week period (see also Forsgren and Humble 2016).

After removing incomplete responses, we had collected 9,292 fully completed surveys out of which 7,522 were from technical professionals (the remainder were from participants from other functions, such as project management, sales, and marketing). The survey was accessed 15,233 times, thus, the response rate was 61%. Because we were interested in the responses of technical professionals, our analysis focuses only on these respondents, who include IT operators (e.g., network administrators, system administrators, operating system administrators, etc.), developers, security professionals, site reliability engineers, DevOps engineers, etc. They came from 261 unique countries with the highest participation coming from the United States (46.8%), the United Kingdom (7.2%), India (4%), Australia (3.8%), Canada (3.7%), and Germany (3.1%). Most respondents worked in Technology companies (21%), followed by Web Software (11.3%), and Education (7.9%) (Forsgren and Humble 2016).

Our analysis began with an investigation of the data and respondents. First, we compared early and late respondents and found no significant differences. To assess the effect of common method bias, we took several steps. First, an exploratory factor analysis was performed and extracted five factors explaining 61.7% of the variance, with no single factor accounting for significant loading ($p<0.10$) for all items (MacKenzie and Podsakoff 2012). Second, following Lindell and Whitney (2001), we adjusted for the second smallest positive correlation in the data, and all significant correlations remained significant. Therefore, we conclude that common method bias likely did not affect our evaluation of the research model.

Method for the Taxonomy Development

We have conducted cluster analyses to develop the taxonomy of Software Delivery Performance Profiles. Candidate sets of clusters were obtained using five different methods: Ward (1963), between-groups linkage, within-groups linkage, centroid, and median (Forsgren and Humble 2016). We have considered different results with between three and five clusters and have evaluated them based on change in fusions coefficients, imbalances in terms of number of individual settings in each cluster, and univariate F-statistics (Ulrich and McKelvey 1990). According to these criteria, the three-cluster solution that was based on Ward's method performed best, thus it was used for the remainder of the study. The solution included one large cluster with 43.5% of the respondents (Cluster 3) and two smaller clusters with 29.6% and 26.8% of the respondents, respectively (Clusters 1 and 2) (Forsgren and Humble 2016).

Further, we have conducted post-hoc comparisons of the throughput and stability measure means to understand the clusters in terms of the performance profiles they represent, using Duncan's Multiple Range test (Hair et al. 2006). We found significant differences for each measure using pairwise comparisons across each cluster, with similar means within each cluster, but means significantly different ($p < 0.05$) from those of other clusters. That indicates that all variables of Cluster 1 perform significantly higher than those of the other clusters; all variables of Cluster 3 perform significantly lower than those of the other clusters; and all variables of Cluster 2 perform significantly lower than those of Cluster 1 and significantly higher than those of Cluster 3 (Forsgren and Humble 2016). It shall be noted that all measures in Table 1 are based on a 6-point scale, where 6 is the highest and 1 is the lowest performance. Thus, low *lead-time for changes* and low *mean time to restore* are expressed as high values and high *lead-time for changes* and high *mean time to restore* are expressed as low values (Forsgren and Humble 2016). According to the results of the post-hoc comparisons, we have named Cluster 1 as the "High Performers", Cluster 2 as the "Medium Performers", and Cluster 3 as the "Low Performers" (Table 2).

		Cluster 1	**Cluster 2**	**Cluster 3**
Software Delivery Performance Attributes	**F-values**[a]	**High Performers**	**Medium Performers**	**Low Performers**
Throughput Release frequency	5261.372***	4.30 (H[b])	3.57 (M)	2.06 (L)
Lead time for changes	3916.669***	5.27 (H)	3.37 (M)	3.07 (L)
Stability Mean time to restore	741.113***	5.52 (H)	5.32 (M)	4.57 (L)

[a] Significances of the F-values are at the 0.001 level (***)
[b] H, M, and L stand for high, medium, and low cluster means, based on Duncan's Multiple Range Test.

Table 1. Software Delivery Performance Profile Taxonomy
(adapted from Forsgren and Humble 2016)

Table 2 describes the three Performance Profiles. Measures for Throughput and Stability move together across all Performance Profiles. In terms of the EOQ that means that high performers have smaller batch sizes (higher throughput) and low performers have larger batch sizes (lower throughput). Furthermore, there are no trade-offs between throughput and stability (Forsgren and Humble 2016).

Role	**Freq**	**Performance Profile**
High Performers	n=1,879 (29.6%)	Throughput and Stability measures perform at significantly higher levels than those of the other two performance profiles.
Medium Performers	n=2,759 (43.5%)	Significantly lower levels of Throughput and Stability than the High Performers, but also significantly higher levels of Throughput and Stability than the Low Performers.
Low Performers	n=1,700 (26.8%)	Throughput and Stability measures perform at significantly lower levels than those of the other two performance profiles.

Table 2. Software Delivery Performance Profiles
(adapted from Forsgren and Humble 2016)

Conclusion and State of the Complete Study

Drawing on EOQ, we have used throughput and stability as key measures for capturing the performance-related concerns of key stakeholders found in development and operations in many organizations and uses these measures to define Software Delivery Performance. The empirical analysis has resulted in a taxonomy of three distinct Software Delivery Performance Profiles for DevOps settings; the performance clusters demonstrate that organizations achieve throughput and stability concurrently. Thus, it is possible to achieve high throughput and high stability without any tradeoffs.

The second part of the study is going to build on the taxonomy of Software Delivery Performance Profiles. We will investigate the effects of individual DevOps practices and cultural transformations on the Software

Delivery Performance Profiles in different DevOps development settings, using logistical regression analysis. We have developed hypotheses that model the effects of Automated Continuous Integration, Automated Test Suites, Test Coverage, Test Lead Time, Version Control, as well as DevOps Culture on Software Delivery Performance.

REFERENCES

Allspaw, J. 2010. "MTTR Is More Important than MTBF (for Most Types of F)," Kitchen Soap. (https://www.kitchensoap.com/2010/11/07/mttr-mtbf-for-most-types-of-f/, accessed April 28, 2019).

Bird, J. 2012. "Software Development Metrics That Matter," *DZone Agile Zone*. (https://dzone.com/articles/software-development-metrics, accessed April 28, 2019).

Debois, P. 2011. "Opening Statement," *Cutter IT Journal* (24:12), p. 3.

Forsgren, N., and Humble, J. 2016. "DevOps: Profiles in ITSM Performance and Contributing Factors," in *Western Decision Sciences Institute Annual Meeting*, Las Vegas, NV, pp. 234–259.

Hair, J. F., Black, B., Babin, B., Anderson, R. E., and Tatham, R. L. 2006. *Multivariate Data Analysis*, (6th ed.), New Jersey: Pearson.

Harris, F. W. 1913. "How Many Parts to Make at Once," *Factory, The Magazine of Management* (10), pp. 135–136.

Humble, J., and Molesky, J. 2011. "Why Enterprises Must Adopt Devops to Enable Continuous Delivery," *Cutter IT Journal* (24:8), pp. 6–12.

Humble, J., Read, C., and North, D. 2006. "The Deployment Production Line," *Agile Conference, IEEE*.

Karlsson, C., and Ahlstrom, P. 1996. "Assessing Changes towards Lean Production," *International Journal of Operations & Production Management* (16:2), pp. 24–41.

Khan, L. R., and Sarker, R. A. 2002. "An Optimal Batch Size for a JIT Manufacturing System," *Computers & Industrial Engineering* (42:2), pp. 127–136.

Khomh, F., Dhaliwal, T., Zou, Y., and Adams, B. 2012. "Do Faster Releases Improve Software Quality? An Empirical Case Study of Mozilla Firefox," in *IEEE International Working Conference on Mining Software Repositories*, pp. 179–188.

Kilpatrick, J. 2003. "Lean Principles," *Utah Manufacturing Extension Partnership*.

Kullstam, P. A. 1981. "Availability, MTBF and MTTR for Repairable M out of N System," *IEEE Transactions on Reliability* (30:4), pp. 393–394.

Leite, L., Rocha, C., Kon, F., Milojicic, D., and Meirelles, P. 2019. "A Survey of DevOpsConcepts and Challenges," *ACM Computing Surveys Surveys* (52:6), 127:1-127:35.

Lindell, M. K., and Whitney, D. J. 2001. "Accounting for Common Method Variance in Cross-Sectional Research Designs," *Journal of Applied Psychology* (86:1), pp. 114–121.

MacKenzie, S. B., and Podsakoff, P. M. 2012. "Common Method Bias in Marketing: Causes, Mechanisms, and Procedural Remedies," *Journal of Retailing* (88:4), pp. 542–555.

Radigan, D. 2015. "Five Agile Metrics You Won't Hate," Atlassian Agile Coach. (https://www.atlassian.com/agile/project-management/metrics, accessed April 28, 2019).

Reinertsen, D. G. 2009. *The Principles of Product Development Flow: Second Generation Lean Product Development*, Redondo Beach, CA: Celeritas.

Rother, M., and Shook, J. 2003. *Learning to See:Value Stream Mapping to Add Value and Eliminate Muda*, Lean Enterprise Institute.

Tang, O., and Musa, S. N. 2011. "Identifying Risk Issues and Research Advancements in Supply Chain Risk Management," *International Journal of Production Economics* (133:1), pp. 25–34.

Ulrich, D., and McKelvey, B. 1990. "General Organizational Classification: An Empirical Test Using the United States and Japanese Electronic Industry," *Organization Science* (1:1), pp. 99–118.

Wang, X., Conboy, K., and Cawley, O. 2012. "'Leagile' Software Development: An Experience Report Analysis of the Application of Lean Approaches in Agile Software Development," *Journal of Systems and Software* (85:6), pp. 1287–1299.

Ward, J. H. 1963. "Hierarchical Grouping to Optimize an Objective Function," *Journal of the American Statistical Association* (58), pp. 236–244.

Wiedemann, A., Forsgren, N., Wiesche, M., Gewald, H., and Krcmar, H. 2019. "Research for Practice: The DevOps Phenomenon," *Communications of the ACM* (62:8), pp. 44–49.