DEVOPS: PROFILES IN ITSM PERFORMANCE AND CONTRIBUTING FACTORS

Nicole Forsgren
Chef Software
nicole@chef.io

Jez Humble
UC Berkeley, School of Information
jez@jezhumble.net

ABSTRACT

This paper describes IT performance using the Economic Order Quantity (EOQ) model, and presents the key factors of effective information technology (IT) team performance in the context of DevOps and IT service management. It then presents the development of a construct to measure IT team performance in the context of DevOps, based on three factors, two throughput measures (lead time to deploy and deployment frequency) and a stability measure (mean time to recover); these correspond to EOQ variables batch size (measured as deployment frequency) and transaction cost (lead time to deploy and mean time to recover). Based on a sample of 7,522 IT professionals worldwide, we conduct a hierarchical cluster analysis and find that the throughput and stability measures move together, consistent with EOQ. The analysis reveals three IT performance profiles: high, medium, and low. Further analysis shows that innovations can be used to impact these IT performance profiles. Implications for research and practice are discussed.

KEYWORDS: DevOps, Economic Order Quantity model, cluster analysis, IT performance, throughput, stability, transaction costs, IT service management

INTRODUCTION

As IT became a critical business enabler from the 1970s onwards, frameworks for managing the lifecycle of IT services became popular. This was led by ITIL [1]. However, the improved efficiencies and reliability offered by these ITIL best practices were not designed for fast IT service delivery. The software development practices in the first three versions of ITIL were based upon the waterfall or V-model. Software projects delivered using this model typically last many months or years before being released to users [2].

The advent of the internet changed the nature of IT service management (ITSM), bringing complex, internet-scale hosted services that required the ability to build, evolve and operate rapidly changing, secure, resilient systems at scale [3]. Agile software development

provided a suitable approach for the development of these products [3, 4], but traditional ITSM practices were unable to cope with the scale of -- and frequency of changes to -- the operational environment. In the traditional ITSM paradigm, it was believed that shorter lead times and more frequent releases led to poor stability and quality in the operational environment [1].

Thus practitioners experimented with new ways to manage the evolution and operation of web-scale systems, creating a new paradigm of IT service management known as DevOps [5]. A key claim of this movement is that higher throughput *and* higher levels of stability are possible at scale through a combination of cultural change and the application of a number of technical and management practices derived from Lean manufacturing [6]. A further claim is that these higher levels of performance are applicable in domains beyond the web and around the world [7, 8].

This study contributes to the literature in two ways. First, it applies the Economic Order Quantity (EOQ) model to the context of ITSM to identify two dimensions of IT performance in the context of ITSM: throughput and stability. In this effort, we develop an empirical taxonomy of IT performance profiles in ITSM, and find that the population splits cleanly into a high, medium and low IT performers. Our analysis verifies the applicability of EOQ in ITSM; that is, throughput and stability measures "move" together. Second, our study identifies business innovations that may impact IT performance in the context of ITSM and investigates this impact. Our analysis suggests that both throughput and stability improvements are possible when organizations implement short lead times for test infrastructure and approval, use version control for application code and system configurations, and have a trusting organizational culture where information flow is prioritized.

This paper proceeds by presenting the Economic Order Quantity model, which provides the framework for this paper. Building on this model, we then present our IT performance

construct and develop our hypotheses. This is followed by our analysis and results. We then discuss the results and present implications for research and practice.

THEORETICAL FRAMEWORK

Software is typically developed in long-running projects with infrequent releases because the transaction cost of pushing out software to users is very high [e.g., 9]. For example, ITIL Service Transition describes eight major processes spanning multiple functions throughout organizations in order to make changes to services [1]. These processes are thus expensive to operate and have a lead time typically measured in weeks or longer [e.g., 10].

The Economic Order Quantity (EOQ) model was a precursor to Lean manufacturing [e.g., 11, 12, 13], and was derived by was derived Ford W. Harris in 1913 [14]. It models the economic trade-offs involved in deciding the optimal batch size, based on the transaction cost and the holding cost. Given a typical hyperbolic transaction cost and a linear holding cost, the total cost forms a U-curve that makes optimization of batch size relatively straightforward. A graph of the curves is shown in Figure 1.

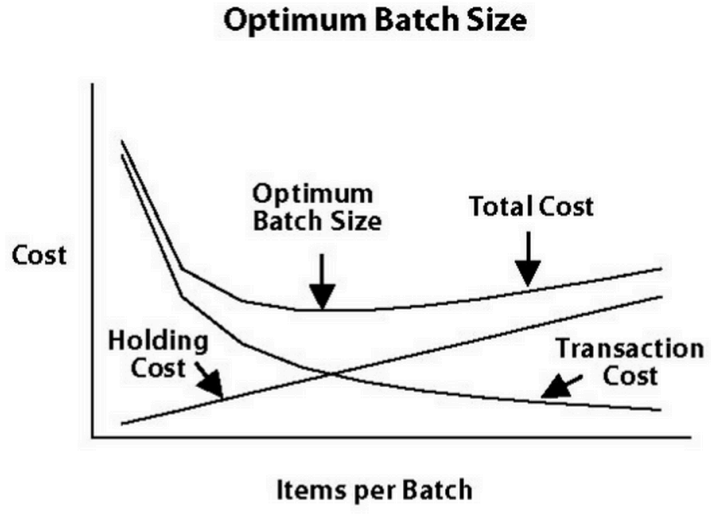

Figure 1. Graph of Economic Order Quantity model [15]

EOQ includes four variables: batch size, holding cost, transaction cost, and total cost [14]. This model was created in manufacturing, so we clarify the meaning of these variables in the context of ITSM. We define a *batch* as a change*;* in ITIL terms, "the addition, modification, or removal of anything that could have an effect on IT Services." The *holding cost* of a batch is the economic opportunity cost of not having the change that constitutes that batch completed and deployed in the production environment. This holding cost can be measured using cost of delay [15]. The *transaction cost* of a batch is the cost of taking the change from version control to production. We also consider a special type of change, wherein the system is degraded and a change must be introduced to restore the system to a good state. In ITIL this is known as an emergency change [1]. Finally, *total cost* is the sum of the transaction cost and the holding cost.

Following EOQ, we assume organizations behave rationally in selecting the optimum batch size based on holding and transaction costs. We assume that holding cost, i.e., the cost of delay, is a pressing need in today's increasingly competitive business environment (as discussed next), and therefore do not include this in our study. Rather, we focus on examining the forces acting on transaction cost and the sensitivity of transaction cost and optimum batch size.

Recently, the economic factors that influence optimum batch size have shifted radically. The holding cost of changes has increased, and the ability to rapidly build and evolve software products and services has become critical to executing business strategy [e.g., 16]. Recent studies from A/B testing demonstrate that 2/3 of experiments demonstrate negative or no business value when released in the form of changes to IT services, and thus the economic opportunity cost of building the wrong thing is potentially very high in strategic products [17]. Therefore, due to these dramatic shifts in holding costs – as seen in the emphasis in speed to market, for example [e.g., 18, 19] – organizations are motivated to reduce batch sizes to get to market faster.

However, following EOQ, as you reduce batch size, transaction costs go up, driving up the total cost of each batch in a nonlinear way. Thus there has been a strong pressure to reduce the transaction costs of working in smaller batches in the context of software delivery.

In IT service delivery, a transaction cost is the lead time for a change, which is most often the work of taking a change from version control to production and is primarily the work of validating that the change will produce the intended effect and will not introduce any defects [20]. This includes, in particular, the cost of integrating, testing, approving and deploying the change, including any associated infrastructure costs such as provisioning new compute, networking or storage resources (whether physical, virtual or cloud-based) [21]. Organizations could reduce the cost of this work directly by omitting all or some of it, but then they run the risk of reducing the quality and stability of their services [22].

Alternatively, innovations can be used to reduce transaction cost. Indeed, a central claim of DevOps is that the use of key practices, or innovations, allows teams to reduce the transaction cost of making changes to services while also improving their stability. This is achieved through several mechanisms. First, the availability of utility compute resources in the cloud computing paradigm has reduced the cost of the infrastructure necessary to make changes and host IT services [23]. However the ability to effectively leverage these resources depends on the second mechanism, known as continuous delivery, which also reduces transaction costs in its own right. Continuous delivery is a set of patterns and practices whereby the infrastructure provisioning and software and infrastructure change validation and transition processes are automated [20]. Both of these mechanisms require the comprehensive use of configuration management and version control to support the management of complex and interconnected infrastructures and services [24]. These mechanisms also require changes in the culture of organizations -- particularly in the

relationship between development, testing and operations functions [25]. Automation and culture change are two important elements of the DevOps paradigm [20, 26].

To summarize, EOQ theory predicts that in today's increasingly competitive business environment, companies are more sensitive to holding costs and therefore will make efforts to reduce transaction costs using innovations. This reduction in transaction costs will decrease batch size, enabling businesses to respond faster to customer needs and market opportunities. This reduction in transaction costs will also result in a reduction in total costs. When we apply EOQ theory in the context of ITSM, we also consider a type of change referred to as "emergency change" in ITIL. This is a change required to fix a critical incident when a service is severely degraded. We measure the lead time to restore service (Time to Restore, or TTR), which is a special case of lead time (and thus transaction cost) in the context of these emergency changes. Short TTR reflects the ability of the organization to achieve high levels of service stability. Therefore, we also expect that, when DevOps practices are deployed correctly, we will see increased throughput, as seen in reduced transaction costs (shorter lead times) and smaller batch sizes (more frequent releases) accompanied by higher rates of stability (reduced transaction costs of emergency changes), and that these will happen in tandem. We call this combination of increased throughput and stability in the context of ITSM *IT performance*.

IT PERFORMANCE

IT Performance in the context of ITSM and DevOps should reflect the concerns of both development and operations departments. Development organizations are typically measured in terms of *throughput* in delivering code, while operations prioritizes the reliability of the infrastructure and *stability* of the services running on this infrastructure [6]. Therefore, following

EOQ, we propose a measure of IT performance that captures both of these dimensions. We outline this below.

Throughput Attributes

When measuring throughput, one can think about software development similarly to a manufacturing plant [21], as stated previously. The first throughput attribute we captured is batch size, which we define as the size of a change that affects IT services. In the case of ITSM, the size of a change is difficult to measure, and is one of the biggest problems in applying Lean manufacturing principles to software development [15]. In fact, batch size has been shown to be difficult to measure, particularly across different contexts [e.g., 27]. Therefore, we proxy batch size using deployment frequency, which is how often code is released into production environments. We see a corollary in just-in-time manufacturing: it is possible to relate batch size and release frequency (production rate), and observe that release frequency tends to infinity as batch size tends to zero [28]. This measure also has the benefit of being highly visible and is often a KPI to software delivery teams [29-31].

The second throughput attribute we captured is lead time for delivery; in software, we consider lead time for changes. We define this as the time it takes for a planned change (either to a service or to infrastructure) to go from being committed to the central version control repository; through integration, testing, and quality assurance; and through to production [32]. Since almost all of the costs in software delivery are fixed, composed of salaries and capital equipment, we use lead time as a proxy for transaction costs. That is, the costs here that can be changed through software development changes related to how quickly a change moves through the system, thereby allowing other changes to also move through the system. This also has the

advantage that it is easier to measure, and can be more easily compared across organizations [e.g., 29, 31].

Stability Attribute

As mentioned above, the ITSM context requires we attend to another type of transaction cost, stability. When measuring stability, we want to capture the lead time of "emergency changes." These are changes necessary to restore the system when a service is severely impaired or completely unavailable. Shorter lead times when delivering emergency changes reflect the ability of the organization to achieve high levels of service stability. Therefore, the stability attribute we collected was mean time to restore (MTTR), which is defined as the time it takes to restore service when an incident occurs and the service becomes unavailable or quality of service is impaired [33]. We selected this metric because it is a good reflection of the transaction cost of emergency changes, and is considered a superior industry standard metric [34].

A simplistic view of IT performance might simply rate performance on a single measure. However, in the context of ITSM and DevOps, we feel it is particularly important to include aspects of IT performance that reflect the two major communities that participate. Therefore, we use a richer, feature-based approach to empirically classify teams into groups based on their reported throughput and stability attributes. Stated formally, our first research question is:

> **RQ1**: When assessing both throughput and stability attributes of performance, what are the different ways in which DevOps teams can demonstrate IT performance through both throughput and stability, and what do these IT performance profiles look like?

IT PERFORMANCE PROFILE

IT Performance includes both throughput and stability measures. The throughput measures were deployment frequency and lead time for changes. The stability attribute was MTTR. ITIL best practices suggest that some software delivery teams can choose to enact

processes that may lead to optimization of some throughput or stability metrics at the expense of others [35]. Seen among DevOps practitioners (e.g., [36, 37]), and following EOQ theory, which suggests that throughput and stability measures will move together as organizations move along the total cost curve, we propose that software delivery teams may achieve IT performance that exhibit other patterns wherein throughput and stability metrics exhibit no tradeoffs and move in tandem. Therefore, we sought to investigate throughput and stability metrics to discover profiles of IT performance seen among software delivery teams.

Using cluster analysis (Appendix A provides details of this analysis), three IT performance profiles emerged based on the three measures of IT performance. These IT performance profiles are now presented based on the results presented in Appendix A; their characteristics are shown below in Table 1.

Table 1: Summary of IT Performance profiles

Role	Freq	Knowledge Exchange Behaviors
High IT Performers	n=1,879 (29.6%)	Achieve the highest throughput of all groups in terms of both deployment frequency and fastest lead times. Restore service from failures the fastest, clearly achieving both throughput and stability.
Medium IT Performers	n=2,759 (43.5%)	See significantly lower levels of throughput and slower lead times than the high throughput group, but higher levels than the Low IT Performers. Have slower MTTR than the high group and faster MTTR than the underperformers. Pretty solidly "middle of the road."
Low IT Performers	n=1,700 (26.8%)	Performs at significantly worse levels than the other two groups across the board, seeing the lowest deployment frequency, the longest lead times, and the longest MTTR.

Consistent with the EOQ model, measures of IT performance moved together across all profiles in our empirical analysis. We observed, consistent with EOQ, that in organizations with lower transaction costs batch sizes are smaller, demonstrated by the high performance group (profile 1) having shorter lead times and higher release frequencies. Conversely, and also

consistent with EOQ, in organizations with higher transaction costs batch sizes are higher, demonstrated by the low performance group (profile 3) having longer lead times and lower release frequencies. Furthermore, we did not observe any trade-offs between lead time and stability. The results of this analysis show that EOQ is a valid model for measuring IT performance in the context of ITSM. Armed with these three emergent IT performance profiles, we now develop the research hypotheses.

FACTORS AFFECTING IT PERFORMANCE PROFILE

DevOps practitioners cite the use of Lean methodologies as a way to change the shape of the total cost of change curve, and hence the optimal batch size, through the reduction in transaction costs either directly or through the use of innovations [15, 38]. The innovations offered by DevOps methodologies would occur in the ITSM and software delivery lifecycles in the stages following development: integration, testing, and acceptance (which we refer to as validation). Once accepted, software is either deployed to production, released to manufacturing, or published, depending on whether the software is part of a service, an embedded system, or a user-installed application (including mobile apps) [39]. When viewed through a lens of ITSM, these innovations can be divided up into two categories: validation (testing and approval), and comprehensive use of configuration management in the form of version control (a key part of what is known as "infrastructure-as-code"). DevOps practitioners [e.g., 40] and prior literature [6, 26] have also highlighted the importance of cultural transformation on achieving good throughput and performance outcomes, so we include culture in our analysis as well. Based on the summary above, our second research question is:

> **RQ2**: How do the validation, version control, and culture innovations of DevOps affect IT performance profiles in the context of ITSM and software delivery?

Next we present the hypotheses of the effects of these innovations on the IT performance profiles.

Validation

First, we will discuss the innovations that support the movement of code through the build, test, and acceptance process. The automation of these processes have been touted as an important innovation among Agile and DevOps practitioners [20] and in the literature [6, 26], and can enable both throughput and stability. Furthermore, build, test and acceptance typically consume a large and often highly variable amount of time in the delivery process [20, 21], so include these innovations in our study.

When developers make changes to their code, these changes must ultimately be validated to ensure they produce the intended effect and do not cause any unintended effects (such as a defect). In the waterfall SDLC, integration and testing are performed after code is "dev complete" [41, 42]. However in the continuous delivery SDLC that forms a component of DevOps, a practice known as continuous integration (CI) is followed [43]. In CI, the integration, build and test process is performed automatically every time an engineer checks code into the central version control repository [20, 44]. When this process fails, developers are notified and fix the problem straight away. When it succeeds, good builds of the software are made available to all downstream users, including testers and IT operations teams. Practitioners [45] and computer science literature [46]suggest CI improves developer productivity. Therefore:

H1: Automated Continuous Integration influences an IT Performance profile, with a continuous integration process that produces good builds daily increasing the likelihood of high IT performance, and lower levels of automated build and testing that produce less frequent good builds increasing the likelihood of low IT performance.

In order for continuous integration to be effective, the automated tests must be trustworthy, and developers must be able to triage failures and fix them rapidly. Trustworthy automated test suites should enable developers to rapidly detect and fix defects, thereby increasing stability. They should also prevent defects from "escaping" the test process, only to be found downstream causing rework; these trustworthy tests therefore increase throughput. Escaped defects must then be found during manual testing cycles, decreasing throughput [47, 48]. Here, we see that automated test suites with good test design contribute to both throughput and stability. Stated formally, we hypothesize:

H2: Trustworthy automated test suites combined with test suite design that makes it easy to reproduce and fix test failures influences an IT Performance profile, with increased levels of trustworthiness and shorter time-to-fix increasing the likelihood of high IT performance, and lower levels of trustworthiness and time-to-fix decreasing the likelihood of high IT performance.

Test coverage is the degree to which the automated tests, be they functional tests or unit tests, protect the software against defects. That is, automated testing is a rigorous and expensive process to implement, and organizations will implement this innovation to differing degrees, either due to cost and time constraints, or just because they have only started their automation efforts. Practitioner reports [49, 50] and prior literature [48, 51] show that more complete test coverage leads to improvements in stability by preventing the escape of defects, and improvements in throughput due to a reduced need for time-consuming manual testing. Therefore, we posit:

H3: The degree of automated test coverage influences an IT Performance profile, with increased levels of automated test coverage increasing the likelihood of high IT performance, and lower levels of automated test coverage decreasing the likelihood of high IT performance.

Lead time for test infrastructure and approval measures the amount of time it takes to get approval for a change to be deployed, and the lead time required to get a test environment. That is, this reflects how long it takes to acquire the technology components necessary to support the work of testing, and acceptance. These processes are typically a significant component of the time it takes to move changes from being committed to version control to being deployed to production, and have been shown to impact throughput [52-54]. Therefore we hypothesize:

H4: Lead time for test infrastructure and approval influences an IT Performance profile, with increased lead time for test infrastructure and approval decreasing the likelihood of high IT performance, and decreased lead time for test infrastructure and approval increasing the likelihood of high IT performance.

Version Control

The importance of version control in DevOps practices has been touted for years [7, 55], and its importance is well known in the computer science community [56, 57]. Version control is the art of maintaining changes to information, whether that information be in application code, infrastructure, or databases [57]. The use of version control has been shown to decrease lead times in software development [58]. Version control has been shown to be important in other areas of software development as well; in particular, prior literature suggests the ability to script the creation of environments using information in version control enables the provisioning of high quality test and production environments (where applicable) more rapidly, which in turn enables shorter lead times [59]. In contrast, where environment creation is a manual process (and therefore hard to repeat), mistakes are often made preparing them, and thus much time is typically consumed triaging and reconfiguring them [59, 60]. Therefore we hypothesize:

H5: The use of version control influences an IT performance profile, with increased use of version control increasing the likelihood of high IT performance, and decreased use of version control decreasing the likelihood of high IT performance.

Culture

Organizational culture has often been shown to influence performance outcomes [e.g., 61, 62, 63], and practitioner reports [7, 8] and prior literature [6, 26] strongly support the importance of culture in DevOps and software delivery performance. In fact, some practitioners claim that DevOps teams cannot succeed without a strong cultural shift [e.g., 25], and is equally important and relevant to both development and IT operations populations. Therefore, we include a measure of culture in our study.

In our study, we use a measure of culture based on organizational typologies outlined by Westrum [64], originally used to predict safety outcomes. We selected this framework because IT operations work environments have been shown to be high risk and complex [65], requiring care and safety, and this is particularly important when increasing code deploys, as in DevOps initiatives. We also selected this framework because it includes collaboration and information sharing among groups, which were both highlighted in previous literature [e.g., 26, 66]. Based on prior literature and reports from practitioners, we posit:

H6: Better organizational culture will influence IT performance profiles, with a better culture increasing the likelihood of high IT performance and a worse culture decreasing the likelihood of high IT performance.

Figure 1 summarizes our proposed research model. A team or organization's IT performance profile is observed in their throughput and stability measures, as described earlier. A team or organization's IT performance profile is then influenced by business innovations. In the context of ITSM and DevOps, these business innovations can fall into three categories:

validation (automation of build and test, automated testing, automated test coverage, and lead time for infrastructure and approval), version control, and organizational culture.

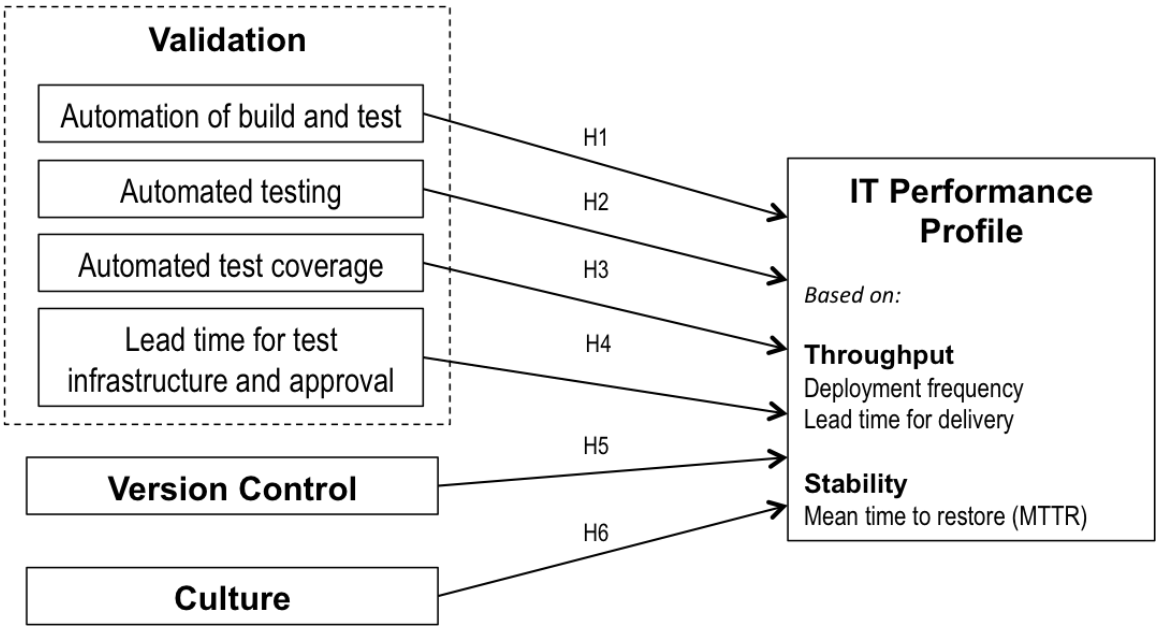

Figure 1. Proposed research model

METHODOLOGY

Data Collection

The data in this study comes from a web-based survey. The population of interest was technical professionals of all specialties involved in DevOps; this could include IT operations, , developers, DevOps engineers, etc. An invitation to participate in the study was posted in various outlets, including professional message boards. After removing incomplete responses, 7,522 surveys were fully completed by technical professionals, with an additional 1,770 surveys completed by professionals in other job functions (such as project management or sales and marketing), yielding 9,292 total completed surveys. In total, the survey was viewed 15,233 times, yielding a response rate of 61% total. Because the population of interest was the technical professionals, the remaining analysis will focus on this group. Of these 7,522 respondents, the

2,826 reported working in IT Operations departments (37.6%), 2,675 reported working in Development or Engineering departments (35.6%), with 1,485 reporting working in DevOps departments. As expected, most respondents came from Technology companies (21%), followed by Web Software (11%) and Education (8%). Note that web software was broken out as a separate industry in the survey, because that specialty in technology is considered the home of the DevOps movement, and often has software development practices that are different than other technology sectors in this regard [67]. Gender was not captured in the survey. **DATA ANALYSIS AND RESULTS**

Assessment of Measurement model

A cross-sectional survey instrument was used, and items for new constructs were developed following Churchill's methodology [68]; items can be requested from the authors. We began by conducting an exploratory factor analysis using principal components extraction and varimax rotation on the independent variables separate from the dependent variables [69]. Any items loading less than 0.60 on their respective construct were dropped, and all cross-loadings were below 0.50. All average variance extracted (AVE) values exceed 0.50 and are greater than their off-diagonal correlations. Finally, the reliabilities of all constructs were over 0.70 [70]. Thus, we can say our measures exhibit good psychometric properties. Factor loading and correlation tables are excluded for length but can be requested from the authors.

Tests of Hypotheses

Because IT Performance profile is a categorical variable, we tested the impact of the business innovations using multinomial logistic regression. The overall model is significant, with a χ^2 of 1087.34 (d.f. = 12, $p < 0.001$). The model is able to predict role with 52.1% likelihood, which is better than 48.08 % by chance, supporting our criteria for classification accuracy. As

recommended, standard errors for all parameter estimates are below 2.0 [71]. Likelihood ratio tests indicate that automation for build and test, automated testing, and automated test coverage do not have a significant effect, thereby not supporting H1, H2, or H3, respectively. Likelihood ratio tests indicate that lead time for infrastructure and approval, use of version control, and organizational culture significantly influence IT Performance profile, supporting H4, H5, and H6, respectively. Results are presented in Table 3.

Table 3. Hypothesis tests

Independent Variables	Chi-squares[a]	Significance
Automation for build and test	2.756	0.252
Automated test	4.525	0.104
Automated test coverage	4.052	0.132
Lead time for infrastructure and approval	682.519	0.000
Version control	41.906	0.000
Culture	13.870	0.001
R^2	0.202[b]	

[a] Chi-squares for likelihood ratio tests are reported for each independent variable, with degrees of freedom = 2 for each independent variable. $\chi^2 = 1055.69$, with 12 degrees of freedom, and significant at $p < 0.001$.

[b] Cox and Snell pseudo R^2 is reported.

Figure 2 shows the emergent model, with three of the six hypotheses supported.

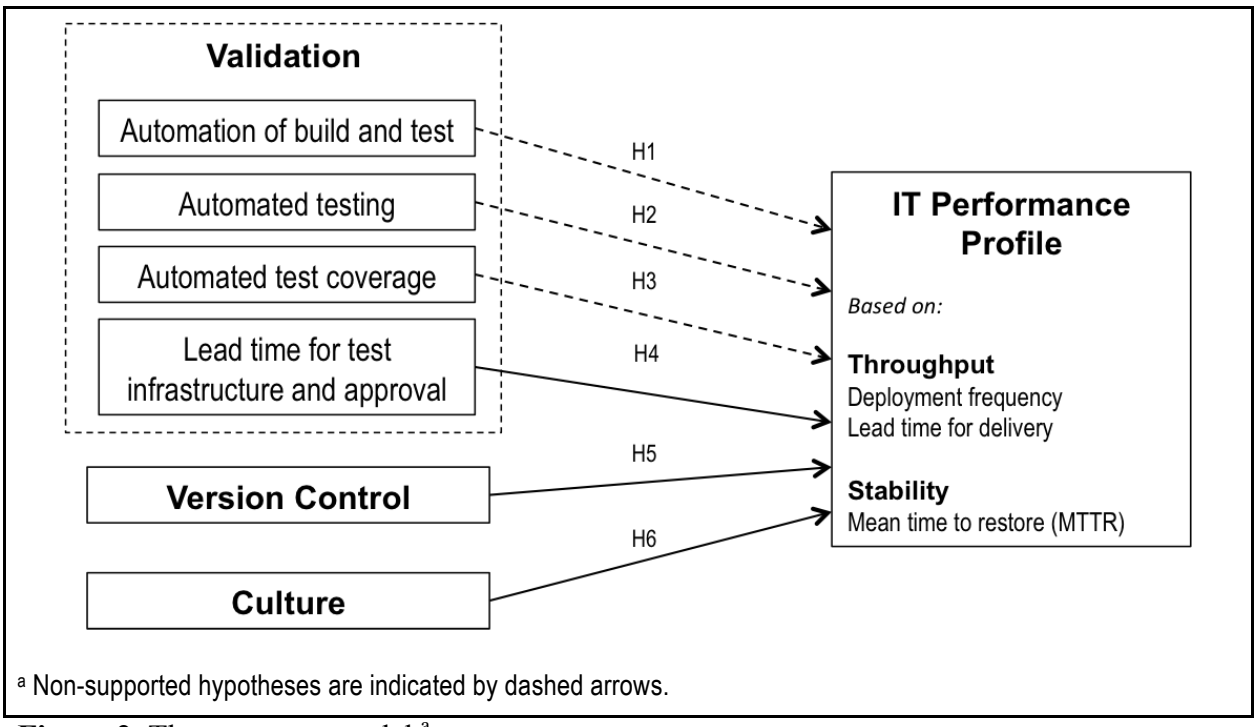

Figure 2. The emergent model [a]

DISCUSSION

This paper makes two contributions. First, it applies the Economic Order Quantity (EOQ) model to the context of software delivery and ITSM. In this effort, it identifies two categories of measures of IT performance, throughput and stability, which also capture the performance-related concerns of the IT populations represented, development and operations. It confirms the applicability of EOQ in this context by developing an empirical taxonomy using cluster analysis and finds that throughput and stability metrics do, indeed, move in tandem. This analysis also generates a taxonomy of three distinct IT performance profiles: high, medium, and low.

Second, EOQ states that throughput and stability (or in terms of EOQ, transaction costs and batch size) can be changed either directly or through the use of innovations. We argue that innovations are a preferable method in this context, and our study identifies three categories of innovations that impact throughput and stability metrics (IT performance profile): validation

(including automation of build and test, automated testing, automated test coverage, and lead time for test infrastructure and approval), version control, and organizational culture. Results indicate that lead time for test infrastructure and approval, version control, and organizational culture have a significant impact on IT performance profile, but that automation of build and test, automated testing, and automated test coverage do not.

Implications for Research

This study has several implications for research. First, it finds support for the EOQ model as a basis of investigating IT performance, positing that throughput and stability metrics may move in tandem. This shows improvements in both throughput and stability are possible without tradeoffs. This method also captures IT performance based on multiple measures, moving beyond just a simple construct. Further research should seek to confirm this finding.

Second, this study found no effects for the use of continuous integration nor comprehensive, reliable test automation, which was quite surprising. Further research is needed to investigate these relationships and processes, since other studies have shown the importance of these elements for achieving high quality software-based products and services [e.g., 72]. Longitudinal analyses may uncover additional insights.

Third, this study found a significant effect of organizational culture on IT performance, confirming the importance of culture in any tech environment, but particularly when information flow is important, such as DevOps. Indeed, this study extends the use of the use of the Westrum model [64] to an additional context, DevOps, positing that information flow was equally important in this context, and finds that a construct of organizational culture based on Westrum's model impacted IT performance. Prior research also suggests that organizational culture is also influenced by the organizational work practices that surround it [e.g., 73, 74, 75], particularly

when engaging with technology [75]. Further research should investigate the reciprocal relationship between the practices and processes undertaken in DevOps transformations and the organizational cultures that take shape.

Implications for Practice

In addition to research, this study has several implications for practice. First, it demonstrates that throughput and stability improvements are observed in DevOps software delivery. Second, this research shows that organizational culture contributes to higher IT performance; that is, higher throughput and shorter time to restore service. Third, we find that the comprehensive use of version control and configuration management has a significant effect on IT performance profile. In many organizations, configuration management of infrastructure and environments is performed manually [76], so this represents an area for improvement.

Limitations

Finally, we discuss the limitations. First, the study's sample may be limiting. The study focused on technical experts who were likely familiar with DevOps, a population relevant to this study but with possible limits to generalization. Second, we identified several innovations that are likely impact IT performance profile, but this list is not exhaustive. Finally, this research is based on a cross-sectional study. While this research design allowed us to capture IT performance profiles and innovations at a point in time, it precluded us from examination or analysis of IT service delivery over time. While our initial investigation is an important one, further research should continue to examine IT performance profiles and additional innovations. Despite these limitations, we believe this research is an important first step in understanding modern software and IT service delivery, and the IT performance profiles that are possible, especially regarding throughput and stability metrics.

ACKNOWLEDGEMENTS

Blinded for review.

REFERENCES

1. Commerce, O.o.G., *Introduction to the ITIL Service Lifecycle*. 2010: The Stationery Office.
2. Barry, E.J., T. Mukhopadhyay, and S.A. Slaughter, *Software project duration and effort: an empirical study.* Information Technology and Management, 2002. **3**(1): p. 113-136.
3. Feitelson, D.G., E. Frachtenberg, and K.L. Beck, *Development and deployment at Facebook.* IEEE Internet Computing, 2013(4): p. 8-17.
4. Ashmore, S. and K. Runyan, *Introduction to Agile Methods*. 2014: Addison-Wesley Professional.
5. Debois, P., *Opening statement.* Cutter IT Journal, 2011. **24**(12): p. 3.
6. Humble, J. and J. Molesky, *Why enterprises must adopt devops to enable continuous delivery.* Cutter IT Journal, 2011. **24**(8): p. 6.
7. Forsgren Velasquez, N., et al., *2014 State of DevOps Report*. 2014, PuppetLabs, IT Revolution Press, ThoughtWorks.
8. *2015 State of DevOps Report*. 2015, Puppet Labs, IT Revolution Press, Pricewaterhouse Coopers.
9. Boehm, B.W., *A spiral model of software development and enhancement.* Computer, 1988. **21**(5): p. 61-72.
10. Hochstein, A., G. Tamm, and W. Brenner, *Service oriented IT management: benefit, cost and success factors.* ECIS 2005 Proceedings, 2005: p. 98.
11. Karlsson, C. and P. Åhlström, *Assessing changes towards lean production.* International Journal of Operations & Production Management, 1996. **16**(2): p. 24-41.
12. Kilpatrick, J., *Lean principles.* Utah Manufacturing Extension Partnership, 2003: p. 1-5.
13. Tang, O. and S.N. Musa, *Identifying risk issues and research advancements in supply chain risk management.* International Journal of Production Economics, 2011. **133**(1): p. 25-34.
14. Harris, F.W., *How Many Parts to Make at Once.* Factory, The Magazine of Management, 1913. **10**: p. 135-136.
15. Reinertsen, D.G., *The principles of product development flow: second generation lean product development*. Vol. 62. 2009: Celeritas Redondo Beach,, Canada.
16. Hagel III, J., et al., *The 2011 Shift Index: Measuring the forces of long-term change.* Deloitte Center for the Edge, 2011. **1**(2011): p. 17.
17. Kohavi, R., et al. *Online experimentation at Microsoft*. in *Proceedings of the Third International Workshop on Data Mining Case Studies, held at the Fifteenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining in Paris, France*. 2009.
18. Chen, J., R.R. Reilly, and G.S. Lynn, *The impacts of speed-to-market on new product success: the moderating effects of uncertainty.* Engineering Management, IEEE Transactions on, 2005. **52**(2): p. 199-212.

19. Lynn, G.S., R.B. Skov, and K.D. Abel, *Practices that support team learning and their impact on speed to market and new product success.* Journal of Product Innovation Management, 1999. **16**(5): p. 439-454.
20. Humble, J. and D. Farley, *Continuous delivery: reliable software releases through build, test, and deployment automation*. 2010: Pearson Education.
21. Humble, J., C. Read, and D. North. *The deployment production line*. in *Agile Conference, 2006*. 2006. IEEE.
22. Berner, S., R. Weber, and R.K. Keller, *Observations and Lessons Learned from Automated Testing.* 2004.
23. Marston, S., et al., *Cloud computing—The business perspective.* Decision Support Systems, 2011. **51**(1): p. 176-189.
24. Fowler, M. and M. Foemmel, *Continuous integration.* Thought-Works) http://www. thoughtworks. com/Continuous Integration. pdf, 2006.
25. Walls, M., *Building a DevOps culture*. 2013: " O'Reilly Media, Inc.".
26. Bang, S., et al. *A grounded theory analysis of modern web applications: knowledge, skills, and abilities for DevOps.* in *Research in information technology*. 2013. ACM.
27. Wang, X., K. Conboy, and O. Cawley, *"Leagile" software development: An experience report analysis of the application of lean approaches in agile software development.* Journal of Systems and Software, 2012. **85**(6): p. 1287-1299.
28. Khan, L.R. and R.A. Sarker, *An optimal batch size for a JIT manufacturing system.* Computers & Industrial Engineering, 2002. **42**(2): p. 127-136.
29. Radigan, D., *Five agile metrics you won't hate*, in *The Agile Coach*. 2015, Atlassian.
30. Bird, J., *Software development metrics that matter*, in *Building Real Software*. 2012.
31. Khomh, F., et al. *Do faster releases improve software quality? an empirical case study of mozilla firefox*. in *Mining Software Repositories (MSR), 2012 9th IEEE Working Conference on*. 2012. IEEE.
32. Rother, M. and J. Shook, *Learning to see: value stream mapping to add value and eliminate muda*. 2003: Lean Enterprise Institute.
33. Kullstam, P.A., *Availability, MTBF and MTTR for repairable M out of N system.* Reliability, IEEE Transactions on, 1981. **30**(4): p. 393-394.
34. Allspaw, J., *MTTR is more important than MTBF (for most types of F)*, in *Kitchen Soap*. 2010.
35. Taylor, S., S. Lacy, and I. MacFarlane, *ITIL: Service transition.* Published by TSO (The Stationery Office), 2007.
36. Allspaw, J. and P. Hammond, *10 deploys per day: Dev & ops cooperation at Flickr*. 2009.
37. Forsgren, N., *DevOps and the Bottom Line*. 2014.
38. Choi, T.Y. and D.R. Krause, *The supply base and its complexity: Implications for transaction costs, risks, responsiveness, and innovation.* Journal of Operations Management, 2006. **24**(5): p. 637-652.
39. Fuggetta, A. *Software process: a roadmap*. in *Proceedings of the Conference on the Future of Software Engineering*. 2000. ACM.
40. Willis, J., *DevOps Culture (Part 1)*, in *DevOps Blog*. ITRevolution Press.
41. Li, E.Y., *Software testing in a system development process: A life cycle perspective.* Journal of Systems Management, 1990. **41**(8): p. 23-31.

42. Balaji, S. and M.S. Murugaiyan, *Waterfall vs. V-Model vs. Agile: A comparative study on SDLC.* International Journal of Information Technology and Business Management, 2012. **2**(1): p. 26-30.
43. Holck, J. and N. Jørgensen, *Continuous integration and quality assurance: A case study of two open source projects.* Australasian Journal of Information Systems, 2003. **11**(1).
44. Ronchieri, E., *A Pattern-based Continuous Integration Framework for Distributed EGEE Grid Middleware Development.* 2004.
45. Shore, J., *The art of agile development: Continuous integration*, in *The art of agile*. 2010.
46. Oram, A. and G. Wilson, *Making software: What really works, and why we believe it*. 2010: " O'Reilly Media, Inc.".
47. Dustin, E., J. Rashka, and J. Paul, *Automated software testing: introduction, management, and performance*. 1999: Addison-Wesley Professional.
48. Koopman, P., et al., *Gast: Generic automated software testing*, in *Implementation of Functional Languages*. 2003, Springer. p. 84-100.
49. Wilson, S., *Don't fear, testing team! DevOps is here!*, in *DevOps.com*. 2014: DevOps.com.
50. Marvin, R., *Testing in a continuous delivery world*, in *Software Development Times*. 2014.
51. Callahan, J., F. Schneider, and S. Easterbrook. *Automated software testing using model-checking*. in *Proceedings 1996 SPIN workshop*. 1996. Citeseer.
52. Schocken, L. and S. Bhat, *Software is not quite ready to eat the world: State of devtst infrastructure survey results*. 2013, Ravello Systems.
53. Whittaker, J., *What is software testing? And why is it so hard?* Software, IEEE, 2000. **17**(1): p. 70-79.
54. Perry, W.E., *Effective Methods for Software Testing: Includes Complete Guidelines, Checklists, and Templates*. 2007: John Wiley & Sons.
55. Wilinski, E., *DevOps best practices: Finding the right tools*, in *New Relic Tech Topics*. 2014, New Relic.
56. Tichy, W.F., *RCS—a system for version control.* Software: Practice and Experience, 1985. **15**(7): p. 637-654.
57. Collins-Sussman, B., B. Fitzpatrick, and M. Pilato, *Version control with subversion*. 2004: " O'Reilly Media, Inc.".
58. Atkins, D.L., et al., *Using version control data to evaluate the impact of software tools: A case study of the version editor.* Software Engineering, IEEE Transactions on, 2002. **28**(7): p. 625-637.
59. Kerravala, Z., *As the value of enterprise networks escalates, so does the need for configuration management.* The Yankee Group, 2004: p. 4.
60. Burgess, T., K. Byrne, and C. Kidd, *Making project status visible in complex aerospace projects.* International Journal of Project Management, 2003. **21**(4): p. 251-259.
61. Homburg, C. and C. Pflesser, *A multiple-layer model of market-oriented organizational culture: Measurement issues and performance outcomes.* Journal of marketing research, 2000. **37**(4): p. 449-462.
62. Berson, Y., S. Oreg, and T. Dvir, *CEO values, organizational culture and firm outcomes.* Journal of Organizational Behavior, 2008. **29**(5): p. 615-633.

63. Gregory, B.T., et al., *Organizational culture and effectiveness: A study of values, attitudes, and organizational outcomes.* Journal of Business Research, 2009. **62**(7): p. 673-679.
64. Westrum, R., *A typology of organisational cultures.* Quality and safety in health care, 2004. **13**(suppl 2): p. ii22-ii27.
65. Velasquez, N.F. and S.P. Weisband, *Work Practices of System Administrators: Implications for Tool Design*, in *CHIMIT 2008*. 2008: San Diego, CA. p. 1.
66. Velasquez, N.F. and S.P. Weisband. *System Administrators as Broker Technicians*. in *CHIMIT 09*. 2009. Baltimore, MD.
67. Willis, J., *The convergence of DevOps*, in *DevOps Blog*. ITRevolution Press.
68. Churchill, G.A., *A Paradigm for Developing Better Measures of Marketing Constructs.* Journal of Marketing Research, 1979. **16**(1): p. 64-73.
69. Straub, D., M.-C. Boudreau, and D. Gefen, *Validation Guidelines for IS Positivist Research.* Communications of the AIS, 2004. **13**: p. 380-427.
70. Chin, W.W., *Issues and Opinions on Structural Equation Modeling.* MIS Quarterly, 1998. **22**(2): p. vii-xvi.
71. MacKinnon, D.P., *Introduction to Statistical Mediation Analysis*. 2008: CRC Press.
72. Karlesky, M., et al. *Mocking the embedded world: Test-driven development, continuous integration, and design patterns*. in *Proc. Emb. Systems Conf, CA, USA*. 2007.
73. Brown, J.S. and P. Duguid, *Organizational learning and communities-of-practice: Toward a unified view of working, learning, and innovation.* Organization science, 1991. **2**(1): p. 40-57.
74. Reiman, T. and P. Oedewald, *Assessment of Complex Sociotechnical Systems–Theoretical issues concerning the use of organizational culture and organizational core task concepts.* Safety Science, 2007. **45**(7): p. 745-768.
75. Orlikowski, W.J., *Using technology and constituting structures: A practice lens for studying technology in organizations.* Organization science, 2000. **11**(4): p. 404-428.
76. Limoncelli, T.A., S.R. Chalup, and C.J. Hogan, *The Practice of Cloud System Administration: Designing and Operating Large Distributed Systems*. Vol. 2. 2014: Pearson Education.
77. Lorr, M., *Cluster Analysis for Social Scientists*. 1983, San Francisco, CA: Jossey Bass.
78. Ward, J.H., *Hierarchical Grouping to Optimize an Objective Function.* Journal of the Americal Statistical Association, 1963. **58**: p. 236-244.
79. Ulrich, D. and B. McKelvey, *General Organizational Classification: An Empirical Test Using the United States and Japanese Electronic Industry.* Organization Science, 1990. **1**(1): p. 99-118.
80. Hair, J.F., et al., *Multivariate Data Analysis (2nd ed.)*. 2006, New Jersey: Pearson Prentice Hall.

Appendix A: Generation of Empirical Taxonomy of IT Performance Profiles

We used cluster analysis to generate an empirical taxonomy of IT performance profiles seen among software delivery teams. Cluster analysis is a technique commonly used to generate

empirical taxonomies from large amounts of data [77]. Cluster analysis was conducted using five methods: Ward's [78], between-groups linkage, within-groups linkage, centroid, and median. Results for solutions with three to five clusters were considered, and were compared in terms of the following: (a) change in fusion coefficients, (b) number of individuals in each cluster (with strongly imbalanced solutions excluded), and (c) univariate F-statistics [79]. Based on these criteria, the solution with three clusters using Ward's method performed best and was selected for further use in our study. This solution included one large cluster (Cluster 3, n=2,759, 43.5% of the respondents) and two smaller clusters (Cluster 1, n=1,879, 29.8% of the respondents; and Cluster 2, 1,700, 26.8% of the respondents).[1]

Post hoc comparisons of the means of the throughput and stability measures were conducted to interpret the IT performance profiles. These comparisons were conducted using Duncan's Multiple Range test [80]. In this analysis, pairwise comparisons are done across each cluster for each measure, and significant differences are reported, wherein the mean for each variable is similar within each cluster, but significantly different ($p < 0.05$ in this study) from the means of other clusters. For example, as reported in Table 1, the deployment frequency of Cluster 1 is significantly higher than that of both cluster 2 and cluster 3, and the deployment frequency of Cluster 2 is significantly lower than that of Cluster 1, and significantly higher than that of Cluster 3. The IT performance profiles were named based on the results of these *post-hoc* tests. Note that in all cases, higher numbers (or responses to the survey questions) indicate more favorable performance. The survey questions and their responses are shown below in Table 2.

Table A1: Differences in throughput and stability measures across IT Performance profiles

		Cluster 1	Cluster 2	Cluster 3
IT Performance measures	**F-values[a]**	**High Performers**	**Medium Performers**	**Low Performers**
Throughput				
Deploy 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 (L)	4.57 (L)

[a] The significance levels of F-values are indicated as follows: *** 0.001 level

[b] H, M, and L indicate that the mean for the cluster was high, medium, or low, respectively, based on Duncan's Multiple Range Test.

[1] 1,184 responses could not be classified due to non-response on one of the classifying variables.