THE ROLE OF CONTINUOUS DELIVERY IN IT AND ORGANIZATIONAL PERFORMANCE

Nicole Forsgren
Chef Software
nicole@chef.io

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

ABSTRACT

This study investigates the impacts of continuous delivery practices in organizations. Continuous delivery is a set of practices designed to optimize the process of taking changes from version control to production or release to manufacturing. Key elements include comprehensive use of version control, automation of the test and deployment process, and the application of continuous integration to rapidly validate the correctness of every change through running the automated build and test process. When viewed through the lens of organizational capabilities theory, CD is an inside-out spanning process. We propose that the use of CD in organizations affects factors that impact the tech workforce today (e.g., perceived burnout and deployment pain), and software delivery performance (e.g., change fail rates and IT performance). We also propose that it indirectly influences organizational performance through IT performance (which is itself a spanning process). We empirically test our model with survey data collected from 4,976 respondents around the world, and find that our hypotheses are supported. The paper's implications for research and practice are discussed.

KEYWORDS: DevOps, continuous delivery, test automation, partial least squares, IT performance, capabilities theory, organizational culture, deploy pain, burnout

INTRODUCTION

Modern organizations that treat software development as a competitive advantage face a problem: they need to deliver software ever faster while preserving the quality and stability of the systems they rely on [1]. A set of technical practices has evolved to meet this need, which together have been popularized under the term *continuous delivery* (CD) [2].

Continuous delivery is typically comprised of a few key processes [2]. In one, developers break down their work into small updates or changes that are made on trunk or mainline in *version control*. Every time the developers commit a change, they get rapid feedback within minutes from *automated tests,* another key aspect. Automated testing allows for fast feedback and learning, providing developers an opportunity to fix any problems immediately when the automated tests fail. Together, these practices are known as *continuous integration*. When the build and tests pass, downstream processes such as comprehensive automated acceptance testing, manual exploratory testing, and performance tests are

triggered. These rely on the ability to deploy the software automatically to an environment created automatically from system and application configuration information stored in version control (known as *deployment automation*). The goal is to take activities such as integration and comprehensive testing that are traditionally performed after a release is "dev complete," and build them into the software development process. This is intended to remove the need for stabilization and hardening phases.

Practitioners report that continuous delivery practices result in higher quality software, lower risk releases, and happier teams [e.g., 3, 4]. The DevOps movement, which has embraced continuous delivery, emphasizes (in addition to the technical practices) the importance of creating a culture of close collaboration between functional silos in order to be able to implement continuous delivery effectively [3-6]. Furthermore, high performing software-powered organizations such as Amazon, Google, and Facebook report extensive adoption of continuous delivery [e.g., 7], suggesting that these technical practices are an important contributor to organizational performance.

This research investigates the role of continuous delivery on software delivery and organizational performance, and the key processes that make up this important capability. This paper proceeds as follows: first, we outline our theoretical framework using capability theory, then develop our hypotheses. Next, we present our methodology, which includes data collection and measurement of research variables. This is followed by analysis and results, and the paper's conclusion.

THEORETICAL FRAMEWORK

Capability theory provides a framework through which to investigate the capabilities of organizations, and the impacts of those capabilities [8]. Capabilities, which have been defined as "organizational and strategic routines by which firms achieve new resource configurations," [9, p. 1107] allow organizations to leverage the resources they have [10]. Based on our prior discussion of CD, it meets the definition of a capability because it combines resources and routines (e.g., programming labor and tooling) in strategic ways. In addition, it is through leveraging these resources that organizations impact their performance [11].

Capabilities theory defines three broad types of capabilities: inside-out, outside-in, and spanning [8]. Inside-out capabilities include technology development, and their goal is to improve efficiency and effectiveness within the firm [8]. As a technology development initiative, CD is an inside-out process because it focuses on internal-facing processes with a goal of increasing efficiency and effectiveness. Specifically, the goal of CD is to increase efficiency and effectiveness of the software delivery process in an organization.

In addition to the more traditional impacts on efficiency and effectiveness that follow from inside-out capabilities such as CD -- seen in the software delivery operationalizations referenced above -- we also include in our investigation perceived impacts of CD: deployment pain and burnout. Increased attention has been paid to the effects of work processes on perceived work outcomes and burnout [e.g., 12], and therefore we include this exploratory analysis in our study.

A second type of capability, spanning capability, is used to link inside-out capabilities and outside-in, or external-facing capabilities [8]. In the context of this research, a spanning capability is seen in IT performance and change failure, which are operationalizations of the software delivery capabilities of an organization. That is, it is a software delivery-specific example of new product development, a spanning process highlighted by Day [8]. Spanning capabilities impact external measures of performance, and provide a mechanism through which inside-out capabilities can impact these external measures.

Hypothesis Development

Referencing capabilities theory [8] and placed in the context of software delivery using DevOps methodologies and practices, we investigate the factors that make up continuous delivery capabilities in organizations. We then examine the impacts of continuous delivery capabilities on other organizational capabilities and processes. These relationships are studied because of the evidence we see in industry [e.g., 2, 3, 13].

We begin our model with an examination of the factors that make up continuous delivery practices. As an inside-out organizational capability, continuous delivery is an internal process with a focus on work that happens within the organization [8]. As such, it combines internal processes that

contribute to effectiveness and efficiency. Prior literature suggests that four factors combine and contribute to this practice: comprehensive use of version control [14], test automation [15], deployment automation [2], and continuous integration [16]. Therefore, we hypothesize:

H1a. Version control is a contributor to continuous delivery.
H1b. Test automation is a contributor to continuous delivery.
H1c. Deployment automation is a contributor to continuous delivery.
H1d. Continuous integration is a contributor to continuous delivery.

Inside-out capabilities, such as continuous delivery, impact spanning capabilities in organizations [8]. Spanning capabilities are those that span internal and external focused work, such as new product development or service delivery [8]. This includes software delivery to customers, or in the context of DevOps, the quality of software delivery to customers. That is, software is being purchased less often off-the-shelf in boxes and is instead delivered continuously through the web or through continuous releases of software [17]. Therefore, it is more meaningful to capture software delivery through quality metrics, which we refer to as IT Performance (which is defined as the ability to deliver software quickly and reliably) and failure rates of changes in software shipped. Stated formally, and following capabilities theory, which posits that inside-out capabilities will have a positive business impact on spanning capabilities:

H2a. Continuous delivery impacts IT performance.
H2b. Continuous delivery negatively impacts change failure rates.

Information systems research states that perceived interaction and usage variables are also important to examine when investigating technology use [e.g., 18, 19]. In the context of software delivery and DevOps, perhaps the most salient moment is found when deploying code [20]. For this reason, we include deployment pain, which is defined as the degree to which code deployments are feared or are perceived as disruptive. Following capabilities theory, which posits that inside-out capabilities will have a positive business impact on work, we posit:

H3. Continuous delivery processes decrease the amount of deployment pain perceived.

In addition to its impact on deployment pain, software delivery and deployment work is often characterized by prolonged stress [21, 22], which can lead to burnout [23]. Indeed, work in technology is

quite demanding and reports of burnout have been increasing, with ripples being felt through the DevOps industry [e.g., 24]. However, the use of work practices that increase effectiveness and efficiency may decrease burnout. Therefore, we include it in our study:

H4. Continuous delivery processes decrease the levels of burnout.

Moving on to our spanning process, IT performance, capability theory suggests that these spanning capabilities can impact organizational outcomes, particularly as they become distinctive capabilities that drive value for an organization [8]. In the context of DevOps and our research model, as organizations and technical teams improve their ability to deliver software quickly and reliably, they are able to deliver value to the customer, thereby positively impacting the organization's goals. Stated formally:

H5. IT performance positively impacts organizational outcomes.

Finally, we include a measure of organizational culture in our study because it has been shown to be significant in prior literature [e.g., 25, 26, 27], and is highlighted in capabilities theory as an important factor in organizations [8]. In fact, a company's culture can make a their capabilities distinctive, thereby improving their value both internally and externally [8]. In this study, we include a measure of organizational culture that highlights trust and information flow, aspects that are key to software delivery in DevOps [22, 28, 29]. Therefore, we hypothesize:

H6a. Organizational culture positively impacts IT performance.
H6b. Organizational culture positively impacts organizational performance.

We also investigate the impact of organizational culture on burnout, because prior literature suggests that good organizational cultures can help decrease feelings of burnout [30, 31]. In particular, we posit that organizational cultures that are high in information flow and trust can decrease levels of burnout felt by those working in high stress environments like DevOps. Stated formally:

H7. Organizational culture high in trust and information flow negatively impacts burnout.

A summary of the proposed model is presented in Figure 1 below.

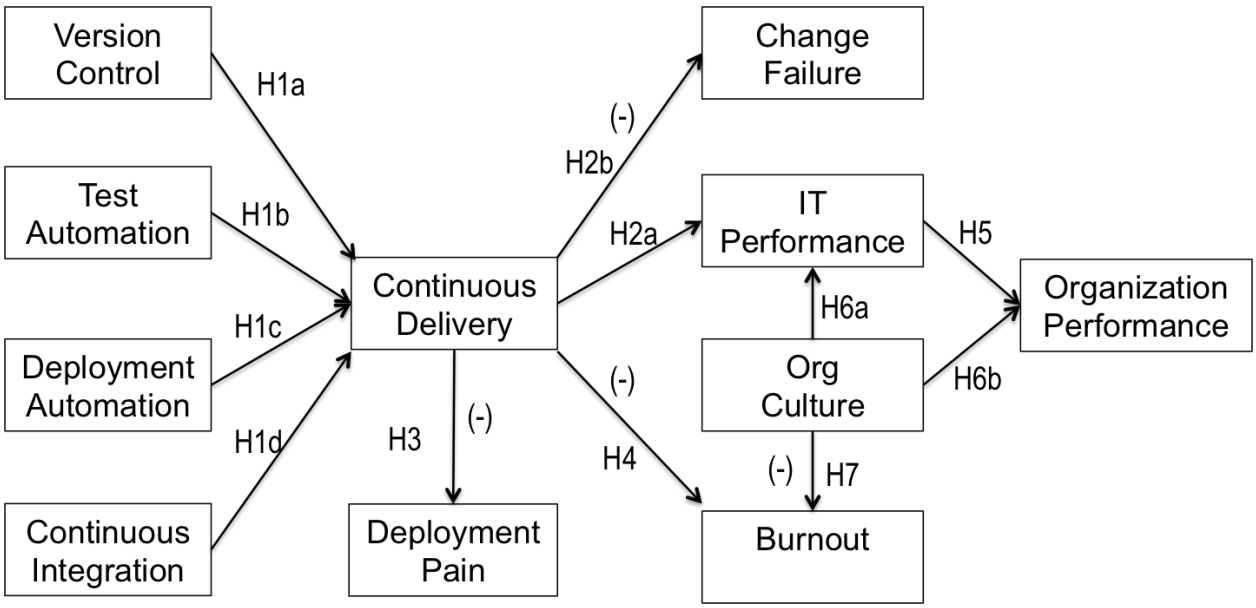

Figure 1. Proposed research model

METHODOLOGY

Data Collection

This study collected data from a web-based survey, targeting professionals familiar with DevOps. Because technical areas similar to this specialty area are still difficult to identify through traditional means [e.g., 32], snowball sampling methods were used. An invitation to participate in the study was posted on social media and related professional message boards; participants were also invited to refer colleagues who worked in and are familiar with DevOps to participate in the study. A web-based survey was utilized because those taking the survey are familiar with technology, and it eased distribution of the survey. The survey was available over a six-week period and could be taken at a time convenient to the respondents.

After removing incomplete responses, 4,976 surveys were fully completed. Of the respondents, 35% reported working in IT Operations departments, 27% reported working in Development or Engineering departments, 19% reported working in DevOps departments, 5% reported working as a consultant, and the remainder reported working in other job functions. Industries included Technology (22%), Web Software (9%), Banking and Finance (8%), Education (8%), Telecommunications (6%),

Consulting (6%), Entertainment or Media (6%), Government (5%), Healthcare (4%), Retail (3%), and Other (23%). It should be noted that the survey separated Technology and Web Software as separate industries, because the DevOps movement started in the web software space, and so these two areas are notably different for the respondents of the survey [6].

Measurement of Research Variables

A cross-sectional survey instrument was used, and items for new constructs were developed following Churchill [33]. First, items were created based on prior literature and talks in industry [e.g., 34, 35, 36]. Next, a card-sorting task was used to test the face and discriminant validity of the items; five technology professionals (in the IT operations and development area) were given 3x5 cards with the one item written on each and asked to sort them into piles by logical groups. Each IT performance item was measured with scale items of increasing duration as outlined in Appendix A. All other items were measured on a 7-point Likert-type scale, anchored on 1 = Strongly disagree to 7 = Strongly agree. Based on feedback, wording on some items was slightly modified for clarity. Before implementing the survey, it was pilot tested with 39 professionals working in DevOps or familiar with DevOps practices.

ANALYSIS AND RESULTS

The proposed research model was tested using partial least squares (PLS) analysis. This method was selected for several reasons: First, it is well suited for exploratory analysis and theory building [37-39]. Second, PLS is appropriate for complex models. Our model includes seven first-order constructs and a second-order construct; as item and construct number increases, researchers may obtain poor model fit simply due to the complexity of the model when using covariance-based structural equation modeling (CBSEM) [40]. Third, PLS is particularly well suited for predictive models, while CBSEM is more concerned with model fit [40]; our study emphasizes prediction of the dependent variable and thus PLS is the preferred method. Finally, PLS has been used in prior literature of technology impacts and is thus suitable for the current investigation [e.g., 38, 40-42].

When determining an adequate sample size in conducting analyses in PLS, a rule of thumb is ten times the largest structural equation model [37, 43]. In this study, the largest structural equation is the

continuous delivery construct, with three paths in the measurement model. Our sample size of 4,976 far exceeds the minimum suggested sample size of 30.

Following prior literature and recommended methods, we analyzed the data in two stages [44]. In the first stage, we assessed the measurement model [42], which we present here. We began by conducting an exploratory factor analysis using principal components extraction and varimax rotation on the independent variables separate from the dependent variables [45]. Any items loading less than 0.50 on their respective construct were dropped (suitable for exploratory research) [46], and all cross-loadings were below 0.50. All average variance extracted (AVE) values exceed the threshold value of 0.50 and are greater than their off-diagonal correlations. Finally, the reliabilities of all first order constructs were over 0.70 [47].

Following Wetzels, Odekerken-Schröder [48], we built the second-order continuous delivery construct by relating it to the first order blocks of version control, test automation, deployment automation, and continuous integration. As Chin [40] states, AVE, reliability, and item loadings do not apply for second order formative constructs, and instead interpretation should be based on the weights, which indicate the relative importance of each indicator. Thus, we can say our measures exhibit good psychometric properties. Items are shown in Appendix A; in the interest of space, loadings and correlations are available from the authors.

Next, we tested the structural model. PLS results include significance of construct relationships with beta coefficients and R^2 measures [43]. These indicate how well the data support the hypothesized model, which makes reading and interpreting the results quite similar to linear regression.

H1a through H1d test the contributions that version control, test automation, deployment automation, and continuous integration make to continuous delivery, respectively. These constructs are all significant, though the path coefficient from deployment automation to continuous delivery, H1c, is quite low. (H1a $b = 0.281$, $p < 0.001$; H1b $b = 0.465$, $p < 0.001$; H1c $b = 0.054$, $p < 0.001$; H1d $b = 0.421$, $p < 0.001$) H2a and H2b posit the impact of continuous delivery on the quality of software delivery as captured in the constructs IT performance and change failure, respectively; these hypotheses are

supported (H2a $b = 0.507$, $p < 0.001$; H2b $b = -0.243$, $p < 0.001$). H3 states that continuous delivery practices will decrease deployment pain; this hypothesis also finds support ($b = -0.681$, $p < 0.001$). H4 similarly states that continuous delivery practices will decrease feelings of burnout, which is also supported ($b = -0.180$, $p < 0.001$). H5 argues that IT performance will positively affect organizational performance; this is supported ($b = 0.192$, $p < 0.001$). H6a and H6b posits that organizational culture will impact the organizational capabilities seen in IT performance and organizational performance, respectively. These hypotheses are supported (H6a $b = 0.062$, $p < 0.01$; H6b $b = 0.274$, $p < 0.001$). Finally, H7 argues that organizational culture will also reduce feelings of burnout; this is also supported ($b = -0.463$, $p < 0.001$).

We also report R^2 measures, first focusing on the organizational capabilities examined in this study. Taken together, continuous delivery and organizational culture explain 28.8% of the variance in IT performance. 14.1% of the variance in organizational performance can be explained by changes in IT performance and organizational culture. In addition to these capabilities, continuous delivery explains 5.9% of the variance in change failure rates, 46.4% of the variance in deployment pain, and 31.8% of the variance in burnout.

The results of the emergent model are presented in Figure 2 below.

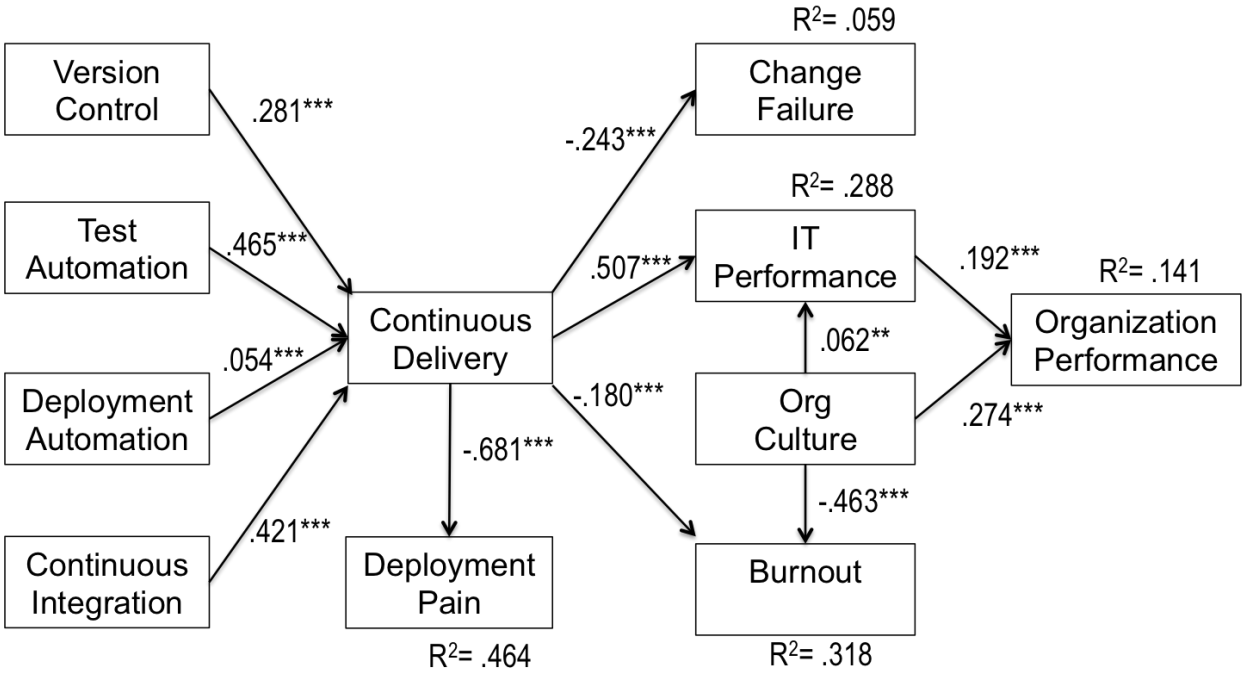

Figure 2. Emergent Model

CONCLUSION

This paper studied the factors that contribute to continuous delivery and found that version control, test automation, deployment automation, and continuous integration are all significant aspects of continuous delivery, although deployment automation plays a much smaller role than the other three. Using capabilities theory, the paper examines the impact of continuous delivery, as an inside-out process, on IT performance, a spanning process; this relationship is supported. The study also investigated the impacts of continuous delivery practices on another quality metric, change fail rates, and finds support for this, though the variance explained is quite low. The study also investigated the extended impact of CD through IT performance on organizational outcomes, together with organizational culture, and found support for these relationships. Finally, we investigate perceptual measures that affect the workforce: deploy pain and burnout. All paths were significant, showing CD plays a central role in the software delivery processes used in DevOps methodologies.

While this investigation looked into a promising and emergent area, the research is subject to limitations. First, this study was cross-sectional in nature, and therefore may miss longitudinal effects, which could be addressed in future efforts. Second, this study concentrated on people familiar with DevOps methodologies, and therefore may exclude those who are practicing continuous delivery but who are not familiar with the term "DevOps." While this helped to reduce variability in our respondents and increase sharing among our target audience, it may limit generalizability to other contexts. Third, this study examines a specific organizational capability in software delivery: continuous delivery. Future research should investigate other possible inside-out capabilities related to software delivery.

Despite these limitations, this research offers contributions to research and industry. First, it offers an empirical investigation of the factors that make up continuous delivery and suggests that version control, test automation, and continuous integration are the most important aspects to implement in the field. Second, it offers evidence that continuous delivery impacts IT performance in software delivery, a relationship that should be investigated in future research. Third, it suggests that continuous delivery also offers benefits to the technical workforce through decreased deployment pain and burnout, information that could be used to encourage management and individual contributors to continue with the difficult technical transformation required to implement continuous delivery pipelines. Fourth, it suggests that organizational culture is important to both IT performance and decreased burnout, consistent with practitioner reports (cite). However, the coefficient from organizational culture to IT performance is quite low and warrants further investigation. Finally, it presents evidence that IT performance and organizational culture impact organizational outcomes; future research should continue to investigate the relationship between IT performance and organizational outcomes.

ACKNOWLEDGEMENTS

Blinded for review.

REFERENCES

1. Aoyama, M., *Web-based agile software development*. Software, IEEE, 1998. **15**(6): p. 56-65.

2. Humble, J. and D. Farley, *Continuous delivery: reliable software releases through build, test, and deployment automation*. 2010: Pearson Education.
3. Humble, J. and J. Molesky, *Why enterprises must adopt devops to enable continuous delivery*. Cutter IT Journal, 2011. **24**(8): p. 6.
4. Fowler, M. and M. Foemmel, *Continuous integration*. Thought-Works) http://www. thoughtworks. com/Continuous Integration. pdf, 2006.
5. Willis, J., *DevOps Culture (Part 1)*, in *DevOps Blog*. ITRevolution Press.
6. Willis, J., *The convergence of DevOps*, in *DevOps Blog*. ITRevolution Press.
7. Feitelson, D.G., E. Frachtenberg, and K.L. Beck, *Development and deployment at Facebook*. IEEE Internet Computing, 2013(4): p. 8-17.
8. Day, G.S., *The capabilities of market-driven organizations*. The Journal of Marketing, 1994: p. 37-52.
9. Eisenhardt, K.M. and J.A. Martin, *Dynamic Capabilities: What are They?* Strategic management journal, 2000. **21**: p. 1105-1121.
10. Kogut, B. and U. Zander, *Knowledge of the Firm, Combinative Capabilities, and the Replication of Technology*. Organization Science, 1992. **3**: p. 383-397.
11. Henri, J.F., *Management control systems and strategy: a resource-based perspective*. Accounting, Organizations and Society, 2006. **31**(6): p. 529-558.
12. Schaufeli, W.B., A.B. Bakker, and W. Van Rhenen, *How changes in job demands and resources predict burnout, work engagement, and sickness absenteeism*. Journal of Organizational Behavior, 2009. **30**(7): p. 893.
13. Marvin, R., *Testing in a continuous delivery world*, in *Software Development Times*. 2014.
14. Appleton, B. and S. Berczuk, *Software configuration management patterns*. 2003, Addison Wesley, Boston.
15. Andres, C. and K. Beck, *Extreme Programming Explained: Embrace Change*. Reading: Addison-Wesley Professional, 2004.
16. Duvall, P.M., S. Matyas, and A. Glover, *Continuous integration: improving software quality and reducing risk*. 2007: Pearson Education.
17. O'reilly, T., *What is Web 2.0: Design patterns and business models for the next generation of software*. Communications & strategies, 2007(1): p. 17.
18. DeLone, W. and E. McLean, *Information Systems Success: The Quest for the Dependent Variable*. Information Systems Research, 1992. **3**(1): p. 60-95.
19. Seddon, P., *A Respecification and Extension of the DeLone and McLean Model of IS Success*. Information Systems Research, 1997. **8**(3): p. 240-153.
20. Luthy, N., *Guidelines for a pain-free technology deployment*, I. Intelligence, Editor. 2013.
21. Kim, G., *IT Revolution Manifesto*, in *IT Revolution Press*. IT Revolution.
22. 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.
23. Maslach, C., W.B. Schaufeli, and M.P. Leiter, *Job burnout*. Annual review of psychology, 2001. **52**(1): p. 397-422.
24. Willis, J., *Karōjisatsu*, in *DevOps Blog*. 2015, IT Revolution Press.
25. 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.
26. Berson, Y., S. Oreg, and T. Dvir, *CEO values, organizational culture and firm outcomes*. Journal of Organizational Behavior, 2008. **29**(5): p. 615-633.
27. 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.
28. 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.

29. Velasquez, N.F. and S.P. Weisband. *System Administrators as Broker Technicians*. in *CHIMIT 09*. 2009. Baltimore, MD.
30. Weberg, D., *Transformational leadership and staff retention: an evidence review with implications for healthcare systems*. Nursing Administration Quarterly, 2010. **34**(3): p. 246-258.
31. Gifford, B.D., R.F. Zammuto, and E.A. Goodman, *The relationship between hospital unit culture and nurses' quality of work life*.
32. Forsgren, N., et al., *The Integrated user satisfaction model: Assessing information quality and system quality as second-order constructs in system administration*. Communications of the AIS, Forthcoming.
33. Churchill, G.A., *A Paradigm for Developing Better Measures of Marketing Constructs*. Journal of Marketing Research, 1979. **16**(1): p. 64-73.
34. Westrum, R., *A typology of organisational cultures*. Quality and safety in health care, 2004. **13**(suppl 2): p. ii22-ii27.
35. Humble, J. and R. Russell, *The agile maturity model applied to building and releasing software*. 2009, ThoughtWorks White Paper.
36. Liu, Y., L. Chengbo, and L. Wei, *Integrated Solution for Timely Delivery of Customer Change Requests: A Case Study of Using DevOps Approach*. International Journal of U-and E-Service Science and Technology, 2014. **7**(2): p. 41-50.
37. Barclay, C., C. Higgins, and R. Thompson, *The Partial Least Squares Approach to Causal Modeling, Personal Computing Adoption and Use as an Illustration*. Technology Studies, 1995. **2**(2): p. 285-309.
38. Chin, W.W., *Issues and Opinions on Structural Equation Modeling*. MIS Quarterly, 1998. **22**(2): p. vii-xvi.
39. Lohmoller, J.-B., *Latent Variable Path Modeling with Partial Least Squares*. 1989, Heidelberg, Germany: Pysica-Verlag.
40. Chin, W.W., *How to write up and report PLS analyses*, in *Handbook of partial least squares*. 2010, Springer Berlin Heidelberg. p. 655-690.
41. Gefen, D., D.W. Straub, and E.E. Rigdon, *An update and extension to SEM guidelines for administrative and social science research*. MIS Quarterly, 2011. **35**(2): p. iii-xiv.
42. Hulland, J., *Use of partial least squares (PLS) in strategic management research: A review of four recent studies*. Strategic management journal, 1999. **20**(2): p. 195-204.
43. Gefen, D., D. Straub, and M. Boudreau, *Structural Equation Modeling and Regression: Guidelines for Research Practice*. Communications of the AIS, 2000. **4**: p. 1-77.
44. Gefen, D. and D. Straub, *A Practical Guide to Factorial Validity Using PLS-Graph: Tutorial and Annotated Example*. Communications of the AIS, 2005. **16**(5): p. 91-109.
45. Straub, D., M.-C. Boudreau, and D. Gefen, *Validation Guidelines for IS Positivist Research*. Communications of the AIS, 2004. **13**: p. 380-427.
46. Hair Jr, J.F., et al., *A primer on partial least squares structural equation modeling (PLS-SEM)*. 2013: Sage Publications.
47. Nunnally, J.C., *Psychometric Theory*. 1978, New York, NY: McGraw-Hill.
48. Wetzels, M., G. Odekerken-Schröder, and C. Van Oppen, *Using PLS path modeling for assessing hierarchical construct models: Guidelines and empirical illustration*. MIS quarterly, 2009: p. 177-195.

APPENDIX A

Unless indicated otherwise, all items measured on a Likert-type scale, 1=Strongly disagree to 7 = Strongly agree

Version Control

Our application code is in a version control system.

Our system configurations are in a version control system.

Our application configurations are in a version control system.

Our scripts for automating build and configuration are in a version control system.

Test Automation

Developers primarily create and maintain acceptance tests.

When the automated tests pass, I am confident the software is releasable.

Test failures are likely to indicate a real defect.

It's easy for developers to fix acceptance test failures.

Developers share a common pool of test servers to reproduce acceptance test failures.

Developers use their own development environment to reproduce acceptance test failures.

Deployment Automation

(Enter a whole percentage)

For the primary application or service you work on, what percentage of your deployments are automated?

Continuous Integration

Code commits result in a build of the software.

Code commits result in a series of automated tests being run.

Builds and tests are executed successfully every day.

Current builds are available to testers for exploratory testing.

Developers get feedback from the acceptance and performance tests every day.

Change Failure

(Enter a whole percentage)

For the primary application or service you work on, what percentage of the changes either result in degraded service or subsequently require remediation (e.g., lead to service impairment, service outage, require a hotfix, rollback, fix forward, patch, etc.)

IT Performance

For the primary application or service you work on, how long does it generally take to restore service when a service incident occurs (e.g., unplanned outage, service impairment, etc.)?

1 More than 6 months

2 Between one month and six months

3 Between one week and one month

4 Between one day and one week

5 Less than one day

6 Less than one hour

For the primary application or service you work on, what is your lead time for changes (i.e., how long does it take to go from "code committed" to "code successfully running in production")?

1 More than 6 months

2 Between one month and six months

3 Between one week and one month

4 Between one day and one week

5 Less than one day

6 Less than one hour

For the primary application or service you work on, how often does your organization deploy code?

1 Fewer than once per six months

2 Between once per month and once every 6 months

3 Between once per week and once per month

4 Between once per day and once per week

5 On demand (multiple deploys per day).

Burnout

I feel burned out from my work.

I feel exhausted.

I am indifferent or cynical about my work.

I feel like I am ineffective in my work.

My feelings about work negatively affect my life outside of work.

Organizational Performance

Select the number that best indicates degree of conformance to your organization's goals over the past year. (1=Performed well below, 7 = Performed well above)

Overall organizational performance

Overall organizational profitability

Relative market share for primary products

Overall productivity of the delivery system

Increased number of customers

Time to market

Quality and performance of applications

Organizational Culture

On my team, information is actively sought.

On my team, failures are learning opportunities, and messengers of them are not punished.

On my team, responsibilities are shared.

On my team, cross-functional collaboration is encouraged and rewarded.

On my team, failure causes enquiry.

On my team, new ideas are welcomed.

On my team, failures are treated primarily as opportunities to improve the system.