The Impact of DevOps Automation, Controls, and Visibility Practices on Software Continuous Deployment and Delivery

Dr. Hanadi Salameh
Argosy University

ABSTRACT

DevOps is a concept intended to blend development and operations processes to increase the efficiency of the software delivery process and early detection of software problems. The ultimate goal of any DevOps implementation is to achieve continuous delivery and deployment of software solutions and services. As DevOps concepts have not yet matured, it is not well understood among software practitioners. This research provides insights on the three elements of DevOps relevant to automation, visibility, and control practices to identify their impact on continuous delivery and deployment. The study found that automation does not have a significant impact in continuous delivery and deployment, while both visibility and control practices had a significant impact. Having that said, visibility practices have proven to have more positive impact in continuous delivery and deployment. When considering all three elements of DevOps visibility had the most impact, followed by automation practices second, and control practices last.

Keywords: DevOps, Continuous Delivery, Continuous Deployment, Automation, Software testing

1. Introduction

DevOps is a concept intended to blend development and operations processes to increase the efficiency of the software delivery process and early detection of software problems (Huttermann, 2012). DevOps is a mix of processes to improve collaboration and communication among developers and operations to address issues during software development (Huttermann, 2012). DevOps reduces the gap between developers, operations and end users, and allows earlier issue detection (Erich et al., 2017). Software and business organizations have started applying DevOps during development process; 44% of surveyed organizations have reported having plans to
embrace DevOps (Stackpole, 2015). DevOps focuses mostly on aspects of automation of software development, testing, and deployment processes (Callanan & Spillane 2016, Elberzhager et al. 2017, Fazal-Baqiae et al. 2017). In addition, it focuses on instilling more collaboration among software development and IT operation teams, measurement and monitoring of software development and deployment quality parameters and metrics (Callanan & Spillane 2016, Chen 2017, Elberzhager et al. 2017, Schneider 2016). Continuous deployment and delivery are the ultimate target of any DevOps implementation. Continuous delivery is having the ability to do frequent deployment (Fowler, 2013). Continuous deployment means that every software change goes through an automated deployment pipeline and automatically is put into production, leading to frequent several production deployments (Lwakatare et al., 2016).

2. Study Significance and Objectives

Recently, fast and frequent delivery of software updates to customers has become a great indicator of organizations success. This expected delivery trend by customers has been driven by innovative online companies, such as Amazon, Google and Facebook that deliver great customers services at fast speed in response to customers’ demand (Lwakatare, Kuvaja, & Oivo, 2016). This recent trend towards fast and frequent delivery of software updates to customers is referred to as continuous delivery and deployment (Rodrique et al., 2016). As continuous delivery and deployment can put a strain on software development and operation teams, DevOps concepts and principles have emerged with the goal of breaking down silos within software delivery organizations and encouraging cross-functional continuous collaboration to achieve this continuous deployment and delivery of software solutions and services.

The DevOps phenomenon, despite its growing importance in software industry, has several challenges. Some of these challenges are related to the lack of a clear definition in terms of processes, principles, and dimensions (Kerzazi & Admas, 2016; Smeds, Nybom, & Porres, 2015). This lack of clarity and agreement has resulted in a number of problems, tension, and criticisms whether DevOps is about culture, technical solution or, alternatively, an entirely new role within a software development organization (Kerzazi & Admas, 2016). As a result, it is challenging to organizations to identify which aspects of DevOps should they apply or start with to reap the benefits of continuous delivery and deployment.

This study investigates the various dimensions and areas of DevOps that were identified in the literature (Goerd, 2017; Lwakatare, Kuvaja, & Oivo, 2016; Lwakatare, Kuvaja, & Oivo 2015; Fowler, 2013) and provides empirical results regarding the impact and role of each on continuous deployment and delivery. This should give software organizations and practitioners’ insights on those aspects, processes, and elements of DevOps that will lead to maximum value early on as
they work through the lengthy and complex implementation phases of DevOps initiatives within their software development and operations departments.

3. Research Questions, Hypotheses and Model

The research will attempt to answer the below questions

1. What is the impact of DevOps dimension individually on continuous deployment and continuous delivery, respectively?
2. Which element of the DevOps dimensions has the most impact, and leads to better improvements on continuous deployment and delivery?
3. Which software development automation functions are easier to implement?
4. What is the impact of continuous delivery in continuous deployment?
5. Which area of DevOps processes and practices should organizations start with to reap the benefits of continuous delivery and deployment?

Upon the review of literature (Goerdt, 2017; Lwakatare, Kuvaja, & Oivo, 2016; Lwakatare, Kuvaja, & Oivo 2015; Fowler, 2013), the research deducted and aimed at testing the hypotheses stated hereafter. Testing these hypotheses provides empirical results regarding the impact of the various elements and practices of DevOps on continuous delivery and deployment that will be helpful to software organizations and practitioners when implementing DevOps practices to improve their software delivery quality and frequency.

H1: There is no significant impact of DevOps automation practices on software continuous deployment at the level of significance ($\alpha = 0.05$).

H2: There is no significant impact of DevOps automation practices in software continuous delivery at the level of significance ($\alpha = 0.05$).

H3: There is no significant impact of DevOps control practices in software continuous deployment at the level of significance ($\alpha = 0.05$).

H4: There is no significant impact of DevOps control practices in software continuous delivery at the level of significance ($\alpha = 0.05$).

H5: There is no significant impact of DevOps visibility practices in software continuous delivery at the level of significance ($\alpha = 0.05$).

H6: There is no significant impact of DevOps visibility practices in software continuous deployment at the level of significance ($\alpha = 0.05$).
H7: There is no significant impact of software continuous delivery in software continuous deployment at the level of significance (α = 0.05).

H8: There is no significant impact at level of (0.05) of DevOps dimensions (automation, control and visibility) in continuous deployment.

H9: There is no significant impact at level of (0.05) of DevOps dimensions (automation, control and visibility) in continuous delivery.

To empirically test these hypotheses, the following enhanced conceptual model was developed:

![Figure (1): Study Model Based on (Goerdt, 2017; Lwakatare, Kuvaja, & Oivo, 2016; Lwakatare, Kuvaja, & Oivo 2015; Fowler, 2013)](image)

4. Literature Review

4.1. DevOps Definition

DevOps, short for Developer Operations, is about streamlining the way developers and system administrations work together to deploy code faster and with a higher quality (Borgenholt, Begnum, & Engelstad, 2013). DevOps is a set of practices intended to reduce the time between committing a change to a system and placing it into the system’s production environment while
ensuring high quality (Bass, Weber, & Zhu, 2015). DevOps is a development methodology aimed at bridging the gap between development and operations, emphasizing communication, collaboration, continuous integration, quality assurance and delivery with automated deployment (Jabbari, bin Ali, Petersen, & Tanveer, 2016). DevOps is defined as the enactment of agile concept that a team is responsible for every aspect of a product from development to operation (Erich et al., 2016).

Several organizations are aiming at implementing DevOps within their software development and delivery organizations as it brings so many benefits such as Netflix, Amazon and Target (Null, 2015). A common benefit of DevOps implementation is fast releases achieved through shortening release cycle times (Callanan & Spillane 2016, Elberzhager et al. 2017, Fazal-Baqae et al. 2017). Callanan and Spillane (2016) reported a reduction of the average release cycle time from two weeks to one day. DevOps also improves system and software quality, 50% reduction was reported in potential customer impacting problems after a release (Callanan & Spillane 2016). DevOps also improve collaboration between developers and operations. DevOps implementation is also reported to bring about cultural and organizational benefits (Callanan & Spillane 2016, Chen 2017, Elberzhager et al. 2017, Schneider 2016). Developers and operations are observed to collaborate more especially when teams have to manage operations-related tasks (Callanan & Spillane 2016, Elberzhager et al. 2017, Schneider 2016, Chen 2017). Finally, DevOps is reported to help improve deployment time, frequency, mean time to recover, developer productivity, time to market of new software features, and shortening feedback cycle (Callanan & Spillane 2016). According to (Lwakatare et al., 2016), DevOps is a mind-set demonstrated with a set of practices to encourage cross-functional collaboration between teams specifically software development and IT operations within a software development organization, in order to operate robust systems and speed delivery of change.

4.2. Drivers for Organizations Interest in DevOps

Organizations have been motivated to implement DevOps processes to realize the benefits of automating activities related to software development and delivery (Callanan and Spillane, 2016). According to Callanan and Spillane (2016), standardizing and automating software packaging and deployment, mechanisms are the most important focus areas when it comes to adopting DevOps. In addition, the implementation of automatic verification of software deployment after each software deployments across various environments is a major DevOps benefit. Schneider (2016) reported on automated infrastructure management as another driving factor. By employing the practice of infrastructure-as-code (IaC) to make configurations reproducible. The practice of IaC for automatic and repeated acceleration of software application deployment and environment provisioning is central to DevOps implementation (Cito et al. 2015, Elberzhager et al. 2017, Schneider 2016, Tang et al. 2015).

Software and infrastructure monitoring is another factor in organizations interest in DevOps adoption and implementation (Tang et al., 2015). Problems that are encountered in production
have their root causes in the decisions made during the software development process, such as appropriate test coverage and misconfigurations (Bass et al. 2013, Hamilton 2007). Monitoring is done via tools and scripts to predict production issues before they take place or to determine the proper course of action when issues are encountered according to urgency. Monitoring is also done for getting customer feedback on software usage (Elberzhager et al. 2017).

4.3. DevOps Dimensions and Elements

According to (Goerdt, 2017), DevOps has two dimensions; one focuses on the aspects of the software delivery lifecycle, and the other focuses on the capabilities required. As the first dimension concentrates on software delivery it includes build and continuous integration, deployment automation, and release automation. On the other hand, the second dimension includes three capabilities that drive build and continuous integration, deployment automation, and release automation. These capabilities are Automation, Controls, and Visibility. According to (Goerdt, 2017), the build and continuous integration process starts with a build of the source code that is performed either on demand, on a schedule, or automatically triggered when a developer checks in a change into the source control system. Upon a successful build, the resulting build artifacts will feed the deployment process. Deployment automation include infrastructure provisioning, configuration management, environment management, and migrating of build artifacts across test and production environments. Usually releasing an update across environments can be a challenge in terms of defining and tracking all various changes associated with a release which can include infrastructure, middleware, database, and application changes that should be managed together as a whole. Hence, release automation is required to streamline and simplify the release process.

For the capabilities of automation, controls and visibility required as part of the second dimension of DevOps, as explained, automation allows for a repeatable, fast, and scalable processes. In addition, automation increasing process efficiency, sustainability, software quality, and reduces cost. Automation can be used in many forms such as infrastructure management, automated deployment, release pipeline, automated unit, integration and functional tests, and system monitoring. Controls aspects of DevOps call for injecting controls and appropriate governance through delivery and maintenance processes. Visibility can be facilitated through dashboards, reports, notifications, and system or IT logs.

On the other hand, according to (Lwakatare, Kuvaja, & Oivo, 2016), the dimensions of DevOps are collaboration, automation, culture, monitoring, and measurement. Collaboration deals with a set of practices to encourage cross-functional collaboration between development and IT operations teams within a software development organization, in order to operate resilient systems and accelerate delivery of change. This dimension implies increase in the roles and responsibilities between software developers and IT operation personnel for instance developers
under this principle pay closer attention to deployment scripts, configuration files, and automated integration, load, and performance tests.

The culture element focuses on instilling a culture of empathy, shared responsibility, and support among software development teams and operations. This is achieved by making communication between development and operations less formal and encouraging mutual respect and support to work together in a blameless environment and share responsibilities when an issue or incident takes place. This dimension also brings emphasis on using production feedback to drive development decisions, improvements, and changes to the system.

4.4. Challenges Implementing DevOps

As explained thus far, DevOps calls for continuous software delivery via the collaboration among development and IT operation, but the meaning and the mechanism to achieve such is not clear nor straightforward. There are several challenges when it comes to implementing DevOps principles, practices, and processes. The lack of clear agreed upon definition contributes to the confusion. As a result, the concept of DevOps is not well understood as the concept has not yet sufficiently matured (Borgenholt et al., 2013; Bass et al., 2015; Dyck et al., 2015; Jabbari et al., 2016; Erich et al., 2016; Lwakatare et al., 2016; Callanan & Spillane 2016, Chen 2017). Due to this ambiguity, getting the proper support to implement DevOps practices is a challenge.

Another challenge is related to starting with the correct scope concerning DevOps transition (Wahaballa, 2015) and defining the proper metrics to measure and monitor development and operation practices (Amaradri and Nutralapati, 2016). Over emphasis on metrics can have an adverse effect on the development and team morale; reasonable expectations and paced improvements of metrics is key to the success of DevOps implementation (Hussaini, 2014; Saran, 2015). Finally, tools poses another challenge, choosing the appropriate and right tools is difficult specifically when team members do not have relevant experience (Wettinger, Andrikopoulos, & Leymann, 2015; Stackpole, 2015; Kang, Le, & Tao, 2016).

4.5. Continuous Deployment

Continuous deployment refers to continuously deploying a quality system into production environment automatically (Lwakatare et al., 2016; Humble & Farley, 2010). Deployment process automation is key into achieving this. DevOps achieves this continuous deployment automation through automated continuous integration (CI) pipeline. The CI pipeline is an integrated stream that puts great emphasis on the automation of build, test and deployment processes (Lwakatare et al., 2016). It involves continuous development whereby code written and committed to version control is built, tested and installed (deployed) in the production environment. An automated deployment considers and safeguards the management of dependencies, versions and
configurations of application and infrastructure, which is often a challenge when done manually (Leppanen et al., 2015).

4.6. Continuous Delivery

Continuous Delivery (CD) is a software engineering practice in which development teams build and deliver new versions of a software system in a very short period, e.g., a week, a few days, and in extreme cases a few hours (Humble & Farley, 2010). Continuous delivery ensures that software is built, tested, and prepared in such a way that it can be released into production at any time (Flowler, 2013). It focuses on having a software in a production ready state at any time and all the time (Chen, 2015).

An organization is considered to have continuous delivery if the software is deployable throughout its life cycle, the team prioritizes keeping the software deployable over working on new feature, having an automated timely feedback on each software update made on production, and a push-button on demand deployment can be made of any version of the software on production (Fowler, 2013). Continuous delivery provides several benefits and advantages to software organizations such as reducing deployment risks as it allows for incremental small changes of the software to be deployed easily and fast. It also provides a tangible and believable measure to progress done in terms of updating or enhancing a software feature or service.

5. Research Design and Methods

To achieve the study objectives, the analytical quantitative approach was used in order to gather and analyze data and test hypothesis.

5.1. Data Collection and Study Sample

The data was collected using a structured questionnaire distributed digitally to software practitioners on the LinkedIn social media platform geared to professionals. The study population consists of software practitioners who are members of the Software Development Professionals LinkedIn group such as software developers, testers, release managers, project managers, technical and operation support engineers; the group has approximately 15,000 members. An email invitation was sent to 200 members of the group outlining the purpose of the study and steps to complete the online questionnaire. Forty-two members responded to the questionnaire representing 21% response rate.

5.2. Study Instrument

This study uses the analytical descriptive approach through collecting data that is related to the impact of DevOps elements relevant to control, visibility, and automation on continuous delivery and continuous deployment from the perspective of IT professionals and practitioners. In order
to test the hypotheses, a questionnaire was prepared based on the relevant theoretical framework and the literature review (Goerdt, 2017; Lwakatare, Kuvaja, & Oivo, 2016; Lwakatare, Kuvaja, & Oivo 2015; Fowler, 2013). Five point Likert scale questionnaire was used to evaluate the extent and frequency of DevOps processes and practices performed at software organizations at which respondents have worked in. The questionnaire consists of three sections, which are:

- Section One: Demographic Variables: the demographic information was collected with closed-ended questions, through (3) factors. (The domain of experience in terms of software development, years of experience and region of work).

- Section Two: DevOps dimensions and it contains three parts:
  - Part one: Automation (four Statements).
  - Part 2: Control (five Statements).
  - Part 3: Visibility (five Statements).

- Section Three: Continuous deployment and delivery, and it contains two parts:
  - Part 1: Continuous deployment (three Statements).
  - Part 2: Continuous delivery (three Statements).

Reliability

To calculate the stability of an instrument study, the researcher used the equation of internal consistency using test Cronbach’s alpha shown in Table (2) the results of Cronbach alpha values for all variables of the study and identification of generally higher (60%) which is acceptable in the research and studies, which gives the questionnaire as a whole the reliability coefficient ranged between (0.655-0.868), as shown in Table (2).

Table (2)

<table>
<thead>
<tr>
<th>Variables</th>
<th>Statements</th>
<th>Cronbach Alpha</th>
</tr>
</thead>
<tbody>
<tr>
<td>Automation</td>
<td>6-9</td>
<td>0.714</td>
</tr>
<tr>
<td>Control</td>
<td>10-14</td>
<td>0.655</td>
</tr>
<tr>
<td>Visibility</td>
<td>15-19</td>
<td>0.836</td>
</tr>
<tr>
<td>DevOps</td>
<td>6-19</td>
<td>0.868</td>
</tr>
<tr>
<td>Continuous deployment and delivery</td>
<td>24-29</td>
<td>0.753</td>
</tr>
</tbody>
</table>

5.3. Data Analysis Techniques

To answer the study questions and hypothesis that were formulated to examine the impact of DevOps dimensions on continuous deployment and delivery, a Statistical Package for Social
Sciences (SPSS) to analyze the collected data and test the research hypotheses. The following statistical techniques and tests were used in data analysis:

1. Frequencies and percentages to describe demographical variables.
2. Cronbach’s Alpha reliability.
3. Descriptive statistical techniques: these included means and standard deviations.
4. Simple regression was used to test the hypotheses.
5. Multiple regression was used to show the independent variable(s) with the most impact in dependent variable.

The research type scale included five Likert scale as follows:

<table>
<thead>
<tr>
<th>Never</th>
<th>Rarely</th>
<th>Sometimes</th>
<th>Most of the times</th>
<th>Always</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
</tbody>
</table>

Relative importance, assigned due to:

Class Interval = Maximum Class – Minimum Class

Number of Level
Class Interval = \((5 - 1)/3 = 1.33\)

- The Low degree from 1.00 - 2.33
- The Medium degree from 2.34 – 3.67
- The High degree from 3.68 – 5.00

6. Results of The Study

6.1. Descriptive Analysis of Study Variable (Level of DevOps Dimensions)

The researcher used Mean, standard deviation, item importance and importance level of DevOps as shown in Table (3).

Table (3) Mean, SD, Item Importance and Importance Level of DevOps Dimensions

<table>
<thead>
<tr>
<th>No</th>
<th>Dimension</th>
<th>Mean</th>
<th>Std. Deviation</th>
<th>Item Importance</th>
<th>Importance Level</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>Visibility</td>
<td>3.36</td>
<td>1.11</td>
<td>1</td>
<td>Medium</td>
</tr>
</tbody>
</table>
Descriptive analysis of respondents when (n=42) table (3) showed that mean of (DevOps dimensions), ranged between (3.36 – 2.98), and whole dimensions were in the medium level. Dimension of (visibility) earned the highest mean; the second dimension was (Control), and the last dimension was automation.

6.2.(Level of Automation)

<table>
<thead>
<tr>
<th>No</th>
<th>Statements</th>
<th>Mean</th>
<th>Std. Deviation</th>
<th>Item Importance</th>
<th>Importance Level</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>In your teams, a member can provision application deployment environment via self-service</td>
<td>3.05</td>
<td>1.17</td>
<td>1</td>
<td>Medium</td>
</tr>
<tr>
<td>9</td>
<td>Your team has the release pipeline automated and can move releases across dev, test production environments automatically.</td>
<td>3.02</td>
<td>1.37</td>
<td>2</td>
<td>Medium</td>
</tr>
<tr>
<td>6</td>
<td>Your automated tests cover 80% of system core functionality Whenever a build is deployed on the test environment, basic test i.e. smoke test scripts are automatically executed.</td>
<td>2.98</td>
<td>1.18</td>
<td>3</td>
<td>Medium</td>
</tr>
<tr>
<td>7</td>
<td></td>
<td>2.88</td>
<td>1.35</td>
<td>4</td>
<td>Medium</td>
</tr>
</tbody>
</table>

Total: 2.98 0.93 Medium

Participants of the study were in the medium level about the Automation, table (4) showed that statement (8) came in the first place while statement (7) came in the last place with mean of (2.88) which is in the medium level.

6.3.(Level of Control)
Table (5) Mean, SD, Item Importance and Importance Level of Control

<table>
<thead>
<tr>
<th>No</th>
<th>Statements</th>
<th>Mean</th>
<th>Std. Deviation</th>
<th>Item Importance</th>
<th>Importance Level</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>The development team checks in code more than 3.60 once a day with every code change.</td>
<td>3.60</td>
<td>0.96</td>
<td>1</td>
<td>Medium</td>
</tr>
<tr>
<td>14</td>
<td>Movement of code dev, test, and production is 3.38 granted/denied via a controlled approval process. Unit testing is executed automatically by a 3.02</td>
<td>3.58</td>
<td>1.43</td>
<td>2</td>
<td>Medium</td>
</tr>
<tr>
<td>12</td>
<td>continuous integration tool whenever a code is checked in.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>Integration tests are executed automatically at least once a day and before deploying any change to the test environment.</td>
<td>2.67</td>
<td>1.26</td>
<td>4</td>
<td>Medium</td>
</tr>
<tr>
<td>11</td>
<td>Your software builds run automatically more than once a day whenever a developer checks in to the code repository.</td>
<td>2.50</td>
<td>1.27</td>
<td>5</td>
<td>Medium</td>
</tr>
</tbody>
</table>

**Total** 3.03 0.84 1 Medium

Descriptive analysis for the respondents about control among sample of the study showed that the highest Mean value was in favor of paragraph (10). The lowest Mean value was in favor of paragraph (11). The control variable level was medium.

6.4. (Level of Visibility)

Table (6) Mean, SD, Item Importance and Importance Level of Visibility

<table>
<thead>
<tr>
<th>No</th>
<th>Statements</th>
<th>Mean</th>
<th>Std. Deviation</th>
<th>Item Importance</th>
<th>Importance Level</th>
</tr>
</thead>
<tbody>
<tr>
<td>19</td>
<td>In your team, issues resolved/implemented by the release are accessed easily via dashboards, reports or other applications.</td>
<td>3.76</td>
<td>1.38</td>
<td>1</td>
<td>High</td>
</tr>
<tr>
<td>16</td>
<td>In your team, you have all the test results easily accessible 3.62 1.34 2 Medium to make go / no go decision for the release process.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>Approvals requested and granted / denied for a release can be accessed easily via dashboards, reports or other 3.36 1.51 3 Medium applications. Your teams have high visibility of your automated</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
15 software release process with metrics collected across the 3.24 1.32 4 Medium complete release process.
Your team can easily access deployments carried out as applications.
18 part of a release via dashboards, reports or other 2.83 1.56 5 Medium

| Total | 3.36 1.11 | Medium |

And the visibility was in medium level as shown in table (6), the descriptive analysis showed that paragraph (19) came in the first place with mean of (3.76) and standard deviation (1.38). Paragraph (18) came in the last place. The results assure the visibility as a whole was in the medium level, because mean value of this dimension was (3.36).

6.5. (Level of Continuous Deployment)

Table (7) Mean, SD, Item Importance and Importance Level of continuous deployment

<table>
<thead>
<tr>
<th>No</th>
<th>Statements</th>
<th>Mean</th>
<th>Std. Deviation</th>
<th>Item Importance</th>
<th>Importance Level</th>
</tr>
</thead>
<tbody>
<tr>
<td>25</td>
<td>Your team can rollback a failed deployment or rejected build easily and automatically across dev, test, or 4.05 1.13</td>
<td>High production environments.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>Your team deploys software at least once a month across the release pipe (dev, test, production).</td>
<td>4.00 1.08</td>
<td>High</td>
<td></td>
<td></td>
</tr>
<tr>
<td>26</td>
<td>Your software deployments are short, smooth, and without any problems or challenges.</td>
<td>3.62 0.94</td>
<td>Medium</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

| Total | 3.89 0.88 | High |

The descriptive analysis showed that continuous deployment was in the high level as a whole. Paragraph (25) came in the first place while paragraph (26) came in last.
6.6. (Level of Continuous Delivery)

Table (8) Mean, SD, Item Importance and Importance Level of continuous delivery

<table>
<thead>
<tr>
<th>No</th>
<th>Statements</th>
<th>Mean</th>
<th>Std. Deviation</th>
<th>Item Importance</th>
<th>Importance Level</th>
</tr>
</thead>
<tbody>
<tr>
<td>28</td>
<td>The number and severity of issues encountered when delivering software to customer is very low.</td>
<td>3.79</td>
<td>0.65</td>
<td>1</td>
<td>High</td>
</tr>
<tr>
<td>27</td>
<td>Your team delivers software to customers at least once a month without encountering any major issues.</td>
<td>3.74</td>
<td>0.91</td>
<td>2</td>
<td>High</td>
</tr>
<tr>
<td>29</td>
<td>Delivered software has features that are customer visible and valued.</td>
<td>2.83</td>
<td>1.27</td>
<td>3</td>
<td>Medium</td>
</tr>
</tbody>
</table>

Table (8) showed that continuous delivery was in the medium level as a whole, Mean value scored of (3.45) with standard deviation of (1.27). Paragraph (28) came in the first place with mean of (3.79) and standard deviation (0.65). The total mean of continuous deployment and continuous delivery as a whole was (3.67) with standard deviation of (0.68), that assure it is in the medium level from the perspective of the study sample.

6.7. Regression Analysis

To test the study hypotheses, the researcher used simple and multiple regression analysis to ensure the impact of DevOps dimensions on continuous deployment and delivery as shown below:

H1: There is no significant impact of DevOps automation practices in software continuous deployment at the level of significance ($\alpha = 0.05$).

Table (9) Simple regression to ensure the impact of DevOps automation practices on software continuous deployment

<table>
<thead>
<tr>
<th>$R$</th>
<th>$R^2$</th>
<th>$B$</th>
<th>Beta</th>
<th>$F$ value</th>
<th>df.</th>
<th>Sig.</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.329</td>
<td>0.108</td>
<td>0.309</td>
<td>0.329</td>
<td>4.849</td>
<td>41</td>
<td>0.003*</td>
</tr>
</tbody>
</table>

*: significant at level of (0.05)

From table (9) it is observed that there is significant impact of DevOps automation practices on software continuous deployment, hence rejecting the null hypothesis.
H2: There is no significant impact of DevOps automation practices in software continuous delivery at the level of significance (a = 0.05).

**Table (10) Simple regression to ensure the impact of DevOps automation practices on software continuous delivery**

<table>
<thead>
<tr>
<th>R</th>
<th>R²</th>
<th>B</th>
<th>Beta</th>
<th>F value</th>
<th>df.</th>
<th>Sig.</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.094</td>
<td>0.009</td>
<td>-0.065</td>
<td>-0.094</td>
<td>0.356</td>
<td>41</td>
<td>0.554</td>
</tr>
</tbody>
</table>

And the result of table (10) showed that there is no significant impact of DevOps automation practices on software continuous delivery, hence accepting null hypothesis.

H3: There is no significant impact of DevOps control practices in software continuous deployment at the level of significance (a = 0.05).

**Table (11) Simple regression to ensure the impact of DevOps control practices on software continuous deployment**

<table>
<thead>
<tr>
<th>R</th>
<th>R²</th>
<th>B</th>
<th>Beta</th>
<th>F value</th>
<th>df.</th>
<th>Sig.</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.566</td>
<td>0.320</td>
<td>0.590</td>
<td>0.566</td>
<td>18.857</td>
<td>41</td>
<td>0.000*</td>
</tr>
</tbody>
</table>

*: significant at level of (0.05)

Table (11) showed that there is significant impact of DevOps control practices on software continuous deployment, hence rejecting the null hypothesis.

H4: There is no significant impact of DevOps control practices in software continuous delivery at the level of significance (a = 0.05).

**Table (12) Simple regression to ensure the impact of DevOps control practices on software continuous delivery**

<table>
<thead>
<tr>
<th>R</th>
<th>R²</th>
<th>B</th>
<th>Beta</th>
<th>F value</th>
<th>df.</th>
<th>Sig.</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.419</td>
<td>0.175</td>
<td>0.324</td>
<td>0.419</td>
<td>8.498</td>
<td>41</td>
<td>0.006*</td>
</tr>
</tbody>
</table>

*: significant at level of (0.05)

The result of regression confirms that there is significant impact of DevOps control practices on software continuous delivery, hence rejecting the null hypothesis.
H5: There is no significant impact of DevOps visibility practices in software continuous deployment at the level of significance (a = 0.05).

*Table (13) Simple regression to ensure the impact of DevOps visibility practices on software continuous deployment*

<table>
<thead>
<tr>
<th>R</th>
<th>R²</th>
<th>B</th>
<th>Beta</th>
<th>F value</th>
<th>df.</th>
<th>Sig.</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.706</td>
<td>0.498</td>
<td>0.558</td>
<td>0.706</td>
<td>39.659</td>
<td>41</td>
<td>0.000*</td>
</tr>
</tbody>
</table>

*: significant at level of (0.05)

Regression equation model also showed that there is significant impact of DevOps visibility practices on software continuous deployment, hence rejecting the null hypothesis.

H6: There is no significant impact of DevOps visibility practices in software continuous delivery at the level of significance (a = 0.05).

*Table (14) Simple regression to ensure the impact of DevOps visibility practices on software continuous delivery*

<table>
<thead>
<tr>
<th>R</th>
<th>R²</th>
<th>B</th>
<th>Beta</th>
<th>F value</th>
<th>df.</th>
<th>Sig.</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.490</td>
<td>0.240</td>
<td>0.287</td>
<td>0.490</td>
<td>12.638</td>
<td>41</td>
<td>0.001*</td>
</tr>
</tbody>
</table>

*: significant at level of (0.05)

And the Regression test showed that there is significant impact of DevOps visibility practices on software continuous delivery, hence rejecting the null hypothesis.

H7: There is no significant impact of software continuous delivery in software continuous deployment at the level of significance (a = 0.05).

*Table (15) Simple regression to ensure the impact of software continuous delivery on software continuous deployment*

<table>
<thead>
<tr>
<th>R</th>
<th>R²</th>
<th>B</th>
<th>Beta</th>
<th>F value</th>
<th>df.</th>
<th>Sig.</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.571</td>
<td>0.327</td>
<td>0.771</td>
<td>0.571</td>
<td>19.395</td>
<td>41</td>
<td>0.000*</td>
</tr>
</tbody>
</table>
*: significant at level of (0.05)

A positive relationship between continuous delivery and continuous deployment (R) value was (0.571), Beta also = 0.571, hence rejecting the null hypothesis.

H8: There is no significant impact at level of (0.05) of DevOps dimensions (automation, control and visibility) in continuous deployment.

Table (16) Multiple regression test to show the impact of (DevOps) dimensions (Automation, Control, Visibility) on Continuous Deployment.

<table>
<thead>
<tr>
<th>Model</th>
<th>Unstandardized Coefficients</th>
<th>Standardized Coefficients</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Constant)</td>
<td>1.818</td>
<td>.400</td>
</tr>
<tr>
<td>Automation</td>
<td>-.086</td>
<td>.127</td>
</tr>
<tr>
<td>Control</td>
<td>.236</td>
<td>.159</td>
</tr>
<tr>
<td>Visibility</td>
<td>.479</td>
<td>.119</td>
</tr>
</tbody>
</table>

*: significant at level of (0.05)

Multiple regression test showed that there is no statistically significant impact of (Automation and Control) in continuous deployment, (t) values were (-0.676, 1.481) respectively, with significant more than (0.05), on the other hand (visibility variable) has an impact in continuous deployment, as (t) value = (4.045) and it is significant at level of (0.05).

That assure rejecting the null hypothesis for visibility and the impact size was (49.8%), and accept null hypothesis for automation and control.

H9: There is no significant impact at level of (0.05) of DevOps dimensions (automation, control and visibility) in continuous delivery.

Table (17) multiple regression test to show the impact of (DevOps) dimensions (Automation, Control, Visibility) on continuous delivery.

<table>
<thead>
<tr>
<th>Model</th>
<th>Unstandardized Coefficients</th>
<th>Standardized Coefficients</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Constant)</td>
<td>2.690</td>
<td>.315</td>
</tr>
</tbody>
</table>

*: significant at level of (0.05)
Table (17) showed that automation has a negative relationship with continuous delivery as Beta value = (-0.551) (t=-3.828) (sig=0.00). Control and visibility have positive impact in continuous delivery Beta values (0.372, 0.526), (t) values = (2.291, 3.303) and it is significant at level of (0.05). That leads to rejecting null hypothesis and accept alternative. Moreover, to show the most variable impact in continuous delivery, the researcher used stepwise multiple regression as shown:

Table (18) Stepwise multiple regression

<table>
<thead>
<tr>
<th>Model</th>
<th>Variables</th>
<th>R</th>
<th>R Square</th>
<th>Adjusted R Square</th>
<th>F value</th>
<th>Sig.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Visibility</td>
<td>.490*</td>
<td>.240</td>
<td>.221</td>
<td>12.638</td>
<td>0.001*</td>
</tr>
<tr>
<td>2+1</td>
<td>Automation</td>
<td>.625b</td>
<td>.391</td>
<td>.360</td>
<td>12.531</td>
<td>0.000*</td>
</tr>
<tr>
<td>3+2+1</td>
<td>Control</td>
<td>.682c</td>
<td>.465</td>
<td>.423</td>
<td>11.014</td>
<td>0.000*</td>
</tr>
</tbody>
</table>

Stepwise Multiple Regression test is used to determine the importance of each independent variable separately in contributing to the mathematical model that represents the impact of DevOps dimensions in continuous delivery. Table (18) shows that the order of entry independent variables in the regression equation, visibility has occupied the first place with amount (24%) effect in continuous delivery, while the automation with visibility had effect (39.1%) in continuous delivery, the third variable (control) come to impact with previous variables (46.5%) as a total of variance scored. and that assures rejecting the null hypothesis and accepting the alternative.

7. Discussion

The results of the study indicated that automation does not have a significant impact on continuous delivery and deployment, respectively. Automation by itself as in having automated tests, scripted provisioning of environments, and scripted deployment does not lead to continuous delivery and deployment. This aligns with (Humble & Farley, 2010; Fowler, 2013; Claps et al., 2015) as the fundamental aspect in terms of continuous delivery and deployment is to have controlled processes through a defined automated pipeline. This automated pipeline takes care of defining a workflow of the development cycle in terms of code management, build, testing, and
deployment across the different software environments: development, integration, testing, staging, and production (Chen, 2015). Having automation in isolation of continuous integration leads to no benefits in the software development and operation processes. When asked of the areas scripted to achieve automation in the software development and release process, as shown in Figure (2), IaC to provision software and configure software environment came first followed by software builds automation, packaging, and deployment. As for testing, code testing was the most scripted while integration testing was the least scripted.

**Figure (2) Responses for scripted and automated areas in software development and release process**

![Figure (2) Responses for scripted and automated areas in software development and release process](image)

On the other hand, the results of the study showed that both visibility and control practices have a significant impact on continuous delivery and deployment which in turn supports (Goerdt, 2017). Control practices investigated in the study constitute of checking code into the code repository by the development team more than once a day; running automated builds of the code upon code checking into the code repository; running automated unit testing of code upon successful code build, running automated integration tests before deploying code into test environment; and finally movement of the code across the development environments (development, test, and production) is granted/granted via a controlled automatic/manual approval process. As for visibility practices investigated in the study includes having access into metrics relevant to the integration and release pipeline such as builds status, type of code changes within each build, customer issues addressed within each build or release, failed and successful test cases, and failed and successful deployments along with root cause of failure. All these information is accessible to the software development and operation teams via a dashboard and reports as needed. Leveraging these metrics and insights into having an automated or manual go/no-go decision of software deployments across the pipeline.
The results of the study confirmed that there is a positive correlation between continuous delivery and deployment indicating that when continuous delivery practices increase, continuous deployment increases as well. This aligns with (Caum, 2013; Leppanen et al., 2015) as continuous delivery is a sequences of practices aimed at ensuring that code can be rapidly and safely deployed to production by delivering every change to a production-like environment through a software continuous integration pipeline, continuous deployment is the subsequent step of continuous delivery with every code change is deployed to production automatically.

When considering the impact of all three dimensions of DevOps (automation, control, and visibility) in continuous delivery, the results of the study found that all three elements have an impact; however, visibility practices had the most impact, followed by automation practices second, and control practices last. On the other hand, when considering all three elements of DevOps (automation, control, and visibility) impact on continuous deployment, the results of the study indicated that only visibility practices had a significant impact. The results of the study also observed that when considered along with visibility and control practices, automation practices had an inverse relationship with continuous delivery. This can be justified by the cost and overhead automated test has on the overall cost and time of the software development and delivery process (Dudekula et al., 2012). Overdoing automated tests can lead to an overhead on the software development and integration process (Karhu et al., 2009) consequently impacting continuous delivery. When participants were asked of the easiest to script and automate, software builds came first followed by software deployment. As for software testing, an average of 65.9% indicated that automating various types of test was not easy.

Figure (3) Responses for which is relatively easy to automate
8. Conclusion

Results of the study provide empirical insights to software organizations and practitioners into the various practices of DevOps and the impact they have individually and collectively in continuous delivery and deployment. These results are valuable taking in consideration as conveyed in the literature (Goerdt, 2017; Lwakatare, Kuvaja, & Oivo, 2016; Lwakatare, Kuvaja, & Oivo 2015; Fowler, 2013) that there is no unanimous agreement and alignment when it comes to DevOps practices in terms of important and how organizations should go about implementing DevOps initiatives within their software organizations. The results of the study emphasized such confusion among practitioners. This research study aimed at empirically answering the below questions;

1. What is the impact of DevOps diminisions individually on continuous deployment and continuous delivery, respectively?

The results of the study shows that automation has no significant impact on continuous delivery and deployment. Automation by itself as in having automated tests, scripted provisioning of environments, and scripted deployment does not lead to continuous delivery and deployment. Having automation in isolation of continuous integration does not lead to continuous delivery and deployment. The key is to have controlled processes via a defined automated integration pipeline that takes care of defining a workflow of the development, testing, and deployment processes across the different software environments starting from development, integration, quality assurance testing, staging, and production. Automation is crucial building block toward building this continuous integration pipeline. Both delivery and control processes were found to have a significant impact on continuous delivery and deployment.

2. Which element of the DevOps diminsions has the most impact, and lead to better improvements on continuous deployment and delivery?

Both continuous delivery and deployment were impacted more by visibility practices than control practices. Furthermore, the study identified that both controls and visibility practices had more impact on continuous deployment than continuous delivery. As continuous deployment focuses on taking every software change that passes through a controlled continuous integration pipeline automatically, it is logical that these two processes have a big impact there. Having a defined controlled, visible, and measurable processes to deploy software changes to production is crucial to achieve continuous delivery and deployment.

3. Which software development automation functions are easier to implement and more cost effective?

80% of software practitioners participated in the study reported that software builds are the easiest to automate, followed by software packaging and deployment at 51.2% and 43.9%, respectively. According to participants of the study, environments provisioning and infrastructure as a code
was the most difficult to automate. As for software testing, a average of 34.1% reported that software testing is easy to automate. Code level testing or unit testing was the easiest to automate, and integration testing being the most difficult. Performance testing, functional testing, and security testing were at second, third and fourth in terms of ease of automation, respectively.

4. What is the impact of continuous delivery on continuous deployment?

The results of the study indicated that there is a positive correlation between continuous delivery and deployment indicating that when continuous delivery practices increases continuous deployment increases as well.

5. Which area of DevOps processes and practices should organizations start with to reap the benefits of continuous delivery and deployment?

As several software organization and practitioners are aiming to implement DevOps processes and activities to reap the benefits in terms of quick and quality delivery of software solutions and services to customers, the lack of consensus and agreement in term of DevOps definitions, capabilities, and processes adds to the inherited challenge implementing DevOps. The results of the study identified that even though the majority of literature (Goerdt, 2017; Lwakatare, Kuvaja, & Oivo, 2016; Lwakatare, Kuvaja, & Oivo, 2015; Fowler, 2013) has called for automation to be a significant and crucial player for DevOps implementations, it did not have a significant impact when being implemented as an isolated activity or process without being integrated into a continuous integration pipeline. Hence, organizations are advised to focus on DevOps in terms of achieving continuous integration of the software development process. An automated pipeline that starts with automated builds that are automatically triggered with every code change, followed by software testing, and then deployment to a production like environment is key in reaping DevOps benefits. This is good news to software organizations as the study results revealed that automating software builds and deployment are among the easiest functions to automated in the software development and delivery cycle.

Software testing is a very important component of the continuous automated integration pipeline, however, automated testing may negatively impact the success of DevOps implementation when being overdone or mismanaged. Automation testing was reported among the most difficult aspects to automate in the software development process. Defining high number of automated test cases or scripts such as unit, functional, security, performance, or integration tests can negatively impact the ultimate goal of DevOps goal of continuous delivery and deployment. Even though the literature (Dudekula et al., 2012; Karhu et al., 2009) calls for 80% of tests to be automated as a good practice, the results of the study indicated that having high number of automated test scripts will have negative impact on continuous delivery and deployment. 60% of participants of the study reported that their organization have 80% of automated test most and some of the time.
Once continuous integration pipeline is implemented, organizations should not ignore visibility and control practices. For continuous delivery and deployment having a dashboard through which the software and operation teams have visibility to all release and deployment information such as number and type of issues resolved, number of builds done, test cases executed along with dates, time, and status is very important for the ultimate goal of DevOps initiative of continuous delivery and deployment.

As for visibility practices, having access into metrics relevant to the integration and release pipeline via a dashboard is very important in achieving continuous delivery and deployment. These metrics include builds status, type of code changes within each build, customer issues addressed within each build or release, failed and successful test cases, failed and successful deployments along with root cause of failure, and application health metrics on the production environment. All this information should be accessible to the software development and operation teams via a dashboard and reports as needed. Leveraging these metrics and insights into having an automated or manual go/no-go decision of software deployments across the pipeline.

Finally, having control over the defined DevOps processes relevant to the continuous integration pipeline is very important to the success of DevOps implementation. Having disciplined code management and daily checking, build, tests processes was among the most important activity to control. Finally, movement of the code across the development environments (development, test, and production) should be either controlled automatically upon the success of certain conditions such as successful integration, performance, security tests, or granted upon a defined approval process.

9. References


