Journal of Theoretical and Applied Information Technology
30th November 2022. Vol.100. No 22

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

IMPLEMENTING CONTINUOUS DELIVERY IN A FINTECH COMPANY: A CASE STUDY

[1]LEWIS, [2]RIYANTO JAYADI
[1,2] Information Systems Management Department, BINUS Graduate Program - Master of Information Systems Management, Bina Nusantara University, Jakarta, Indonesia 11480
E-mail: [1]lewisloofis@gmail.com, [2]riyanto.jayadi@binus.edu

ABSTRACT

Various experts promote Continuous Delivery (CD) due to this principle's benefits. Even so, implementing CD is difficult to do correctly. This study evaluates how a company adopts and implements the CD principle, the challenges during implementation, and its impact on a company represented as PT XYZ. PT XYZ is a tech company that has adopted CD for several months as an essential principle of the company's development. We conducted a case study on PT XYZ through observation, interviews, and documentation review, which was then analyzed using thematic analysis and descriptive statistical analysis based on data collected by the company. The results showed that PT XYZ implemented several points of CD technical capability well. However, PT XYZ faced several challenges when transforming its development process using the CD principle: no clear CD implementation framework defined by practitioners, the lack of measurement, and the need to balance the application of CD capabilities. Nevertheless, the application of the CD principle shows a positive impact on PT XYZ psychologically for the IT team and affects the organization's performance.
Keywords: *Continuous Delivery, Continuous Integration, Deployment, Pipeline, Software, Development, DevOps*

1. INTRODUCTION

The principle of continuous delivery began to be recognized among practitioners because of the benefits brought by this principle. The principle of continuous delivery is closely related to DevOps culture and is often discussed together [1], [2]. Various practitioners have written that the application of continuous delivery and the practices behind the continuous delivery principle provide rapid feedback to the development team and improve collaboration between the development, testing, and operations personnel responsible for delivery, thereby affecting software quality [3], [4]. In addition, continuous delivery has a significant impact on quality (measured by change fail rate), organizational Information Technology (IT) performance, overall organizational achievement, and employee burnout [5]. Large digital companies such as Facebook have also adopted the principle of continuous delivery, naming their development principle perpetual development [6], [7].

Regarding the impact of continuous delivery on an organization's IT performance, previous studies have proven that continuous delivery has increased an organization's software delivery speed while maintaining application quality [8]–[10]. High-performance IT-based organizations that successfully implement continuous delivery generally release applications 30 times more frequently with 200 times shorter release wait times, 60 times less potential for failure, and 168 times faster service recovery [11]. Several studies have also found a link between job satisfaction and improvements in employee burnout levels with the application of DevOps principles and continuous delivery as part of DevOps [5], [12], [13].

Despite its many advantages, implementing continuous delivery is difficult to do correctly [14]. Although research on DevOps and the principle of continuous delivery is increasing, the literature studies on DevOps and the principle of continuous delivery generally focus on explaining the tools, practices, and the lack of standards or frameworks needed to implement DevOps [15]. Furthermore, not many case studies study the implementation, challenges, and impacts of implementing the continuous delivery principle empirically and thoroughly in a company with software development as its primary business activity.

This paper is a case study that observes the implementation of the continuous delivery

principle in a company that utilizes software development as its primary business activity. We use PT XYZ (name withheld) to represent a company implementing continuous delivery. PT XYZ is an Indonesian fintech company that has seen massive growth since the pandemic due to the increasing financial literacy of the Indonesian people, especially since the pandemic began. As a fintech company that continues to grow, PT XYZ has experienced considerable growth in the number of employees. For example, from the beginning of 2020 to June 2022, PT XYZ experienced tenfold growth in the number of employees, reaching about 1000. As a company engaged in a sector that continues to grow and has fierce competition, PT XYZ needs continuous delivery to keep up with rapid industry changes even though the organization is growing.

Before this research started, some engineers in PT XYZ had some awareness of the CD principle and its benefits. The engineers started improving the existing software development process according to the CD principle based on the team's interpretation. The problem comes when the team tries to validate whether the improvement effort is correct to the CD principle. This study aims to help PT XYZ to evaluate its existing technical capabilities and improvements according to CD principle

This study discusses the implementation of continuous delivery in the company's software development process in detail, starting from the practices and tools used, changes made to improve continuous delivery, challenges in implementation, and the impact of implementing the continuous delivery principle on the organization. Not only focusing on observing the application of practices and tools related to continuous delivery but also showing the impact of continuous delivery on organizations and what things need to be considered when applying this principle. We hope this research will be helpful for organizations seeking to implement continuous delivery into their development practice. This case study evaluates how PT XYZ implements continuous delivery into the organization's software development practice. During this research, we uncovered how PT XYZ implements each essential continuous delivery technical capability listed by Forsgren et al. (2018). This study also observed how PT XYZ overcame challenges when improving technical capabilities. Besides examining the implementation of continuous delivery technical capabilities, we also observed the impact of the technical capabilities implemented by PT XYZ on the organization's tech team performance.

This research question are RQ1: How does the PT XYZ development team implement technical capabilities of continuous delivery? RQ2: How does PT XYZ as an organization deal with continuous delivery implementation challenges? RQ3: What is the impact of continuous delivery implementation to PT XYZ?

2. LITERATURE REVIEW

2.1. Software Development Lifecycle (SDLC)

Margaret Hamilton introduces software engineering as a discipline used to build applications supporting the NASA mission to land a man on the moon [1]. NASA's success made software engineering gain serious attention as a discipline, and the field keeps being developed onward.

Winston Royce made the first documentation about the software development process in 1970. This documentation writes a development process which is now called the waterfall model. Even if Royce introduced this model as a flawed development process [16], developers would often use this model due to its simplicity. However, the simplicity offered by the waterfall model is not enough to describe real software projects, which have many requirement changes over time. Most of the time, software projects tend to be flexible, so people are looking for a software development process that can accommodate the flexibility in real software projects.

2.2. Agile Software Development

The term agile in software engineering was popularized when Agile Manifesto was created by a group of engineers called Agile Alliance [17]. Four values of the Agile Manifesto are written as follows:

- Individuals and interactions over processes and tools
- Working software over comprehensive documentation
- Customer collaboration over contract negotiation
- Responding to change over following a plan

The values written in the Agile Manifesto became the inspiration for many other software development practices, such as XP, Scrum, and Kanban. These practices have something in common: each recognizes that the software development process is an empirical process,

which means people running the same process can produce different or unexpected results [18].

2.3. DevOps

DevOps is a combination of two terms: development and operation. The terms development and operation are closely related to the functions in a company's technology department, and these two functions usually work separately. DevOps then emerged as a concept that combines these two things by utilizing various technologies such as automated development, deployment, and infrastructure monitoring [19]. Apart from being a concept, some view DevOps as a cultural movement [1]. The different interpretations of various experts sometimes create many misinterpretations about the meaning of DevOps [20], but DevOps represents a movement to support the collaboration of various functions or departments to achieve one goal.

The development of DevOps itself is influenced by the development of today's software development culture, such as agile principles, and the dynamics of work and life in the modern era where everyone is interconnected. Agile principles, which teach teams to be flexible to changes and encourage collaboration, are further explored as a development process in the DevOps movement and applied to the entire organization as an organizational dynamic [1].

2.4. Continuous Delivery

Continuous delivery is the process of releasing new software continuously through continuous integration and automated testing [1]. The release process in software development means implementing changes to the application that the user sees or uses.

According to Forsgren et al. (2018), several technical capabilities become essential to implementing continuous delivery. These capabilities are: 1.) Version Control, 2.) Deployment Automation, 3.) Continuous Integration, 4.) Trunk-Based Development, 5.) Test Automation, 6.) Test Data Management, 7.) Shift Left on Security, 8.) Loosely Coupled Architecture, 9.) Empowering Teams to Choose Tools, 10.) Monitoring, 11.) Proactive Notification

Implementing the continuous delivery principle affects organizational performance and the overall corporate culture [5]. Improved corporate culture affects employee satisfaction through improving organizational performance.

3. RESEARCH METHOD

3.1. Company Background

PT XYZ (name not disclosed due to agreement) is a financial technology (fintech) company domiciled in Jakarta, Indonesia. PT XYZ is an Indonesian digital company that develops systems for trading various financial assets. PT XYZ's applications are released in various forms, such as web and mobile applications (iOS and Android). PT XYZ enjoys massive user base growth and transaction numbers due to increasing financial literacy in Indonesia. This growth also affects company size regarding employee count, which has grown ten times from 2020 to 2022.

The development team within the company itself is divided into several teams. Until this research was written, PT XYZ released two financial applications. The first application is a stock trading application, which also serves as a forum for traders inside the application. The other application is intended for the general public who want to invest long-term through financial assets considered more suitable for the wider community, such as mutual funds and bonds. This study focused on the development team that handles mutual fund and bond applications.

Regarding features, the development of mutual fund and bond applications is divided into the customer-facing side and the side that serves the company's internal (back office). Each side has its specific platform team. For customer-facing apps, the available platform teams are web, iOS, Android, and backend teams that handle database and server systems. For internal-facing applications, the platform team consists of the web and backend teams. Each team works closely with the product team to manage and develop the app's features. Each team consisting of a pool of web developers, iOS, Android, backend, and product teams generally handles a single project, feature, or specific case within the app.

3.2. Data Collection

This research observed how the PT XYZ engineering team does the software development process. The data collection took place from January 2021 to June 2022. We used observation, interview, and documentation review to collect data. The data collected was analyzed and then discussed for a conclusion.

3.2.1. Observation

To observe PT XYZ's development practice, we collected and analyzed several data regarding the development process that PT XYZ has

collected. The time range for data collection was available from January 2021 to June 2022, matching the research period. The metrics collected by PT XYZ during these periods are 1.) Deployment frequency, 2.) The number of incidents (from November 2021), 3.) The number of reported bugs (from July 2021).

The development metrics were analyzed in parallel with PT XYZ development practice to gain insight into any relation between PT XYZ development practice and the metrics collected.

This study observed how PT XYZ develops and maintains its application. PT XYZ has a two-week development sprint, and each development sprint ends with a sprint retrospective. This study focused on the PT XYZ web and server-side teams since both teams are the core maintainer of the PT XYZ system. We observed how development is done in PT XYZ through observing these sprint retrospectives. These sprint retrospectives and discussions were analyzed and linked with continuous delivery principles.

PT XYZ has an automated pipeline to help the engineering team maintain software. We had asked for access to the PT XYZ repository to learn about the pipeline process and to understand this pipeline. We analyzed how PT XYZ arranges this automated pipeline by observing the code behind the automated pipeline.

3.2.2. Interview

PT XYZ tech team's management and key employees were interviewed to understand how the continuous delivery principle was implemented in the organization. Several roles were targeted for interview: PT XYZ's Chief Technology Officer (CTO), Product Manager, Tech Lead (equivalent to Line Manager), and senior employee who has experienced the development process before continuous delivery principles are introduced. These people were interviewed in separate sessions, which took approximately an hour.

There were three major topics discussed during the interview. The first topic discussed the opinion of each interviewee about the implementation of continuous delivery technical capabilities, which consists of 11 points [13]. Each point had several measures to help the interviewee decide whether the company indeed implements a point. The second topic dug each technical capability deeper and asked about missing implementations or challenges when deciding to implement a capability. The third topic discussed implementing the continuous delivery principle on the organization's performance.

Table 1 Key Continuous Delivery Capabilities And The Measures Used To Indicate Continuous Delivery Implementation.

Technical Capability	Effectiveness Measure
Version Control	How does PT XYZ use version control for their application code?
	How does PT XYZ use version control for its system configuration management?
	How does PT XYZ use version control to manage application configuration?
	Does PT XYZ keep scripts and configuration in version control?
Deployment Automation	How many manual steps were taken by the engineering team to do application deployment?
	How much percentage of the deployment process has been automated?
	How long does it take to deploy applications?
Continuous Integration	How many percentages of code commits trigger automated build without manual intervention?
	How many percentages of code commits trigger the test suite to run automatically without manual intervention?
	How many percentages of automated builds and tests are running successfully?
	How does the engineering team distribute the build generated from automation?
	How long does it take for the engineering team to receive feedback from acceptance or performance tests for each build?
	How long does the engineering team fixing a broken build take?
Trunk-based Development	How many active branches are available on version control?
	How does the organization manage code freeze?
	How frequently has the engineering team merged their code into the codebase?

	How long does it take to approve code changes?
Test Automation	Who are the people responsible for maintaining the acceptance and unit test
	What is the proportion between bugs found when testing and in production over time?
	How long does it take to fix test failure?
	What is the quantity of automated test failures that represent an actual defect?
	Do test suites run in every pipeline trigger?
Test Data Management	How long does it take for the engineering team to prepare data for the test?
	How many test suites in PT XYZ can be run without rearranging test data?
Shift Left on Security	How many percentages of features undergo security review during the design process?
	How much time does the review add to the development process?
	Do automated tests cover security requirements?
	How many tools are being used by the development team, which is also approved by the security team?
Loosely Coupled Architecture	Does the application architecture choice affect the team development process?
Empowering Teams to Choose Tools	Do teams in PT XYZ feel they are empowered to choose tools?
Monitoring	How does PT XYZ use data from the monitoring system to make a business decision?
Proactive Notification	How does PT XYZ arrange a notification system to identify problems before customers are aware?

3.2.3. Documentation Review

PT XYZ has several documents regarding the organization's software development practice. This paper also analyzed every documentation, graph, and note related to the software development process. These documents are handy to explain several ideas behind specific tools used by PT XYZ to develop or maintain apps, such as the automated pipeline the engineering team has or the scripts supporting the development process.

3.3. Data Analysis

As described in the data collection methods section, this case study collected qualitative information derived from observations and interviews. Therefore, this case study used thematic analysis to retrieve more information from the collected data. This study also used the company's software development statistics to support gathered insight.

3.3.1. Qualitative Analysis

Thematic analysis is a method to find patterns or themes from the data collected [21]. Thematic analysis was chosen to look for patterns or meanings from the results of interviews and observations. The steps of thematic analysis are 1.) Familiarize yourself with the data collected, 2.) Generate initial code, 3.) Search for themes, 4.) Review various potential themes, 5.) Defining and naming themes, 6.) Make a report.

Thematic analysis was conducted with a deductive approach to determine the code and themes from the research data. In a deductive approach, we view the data through an existing theoretical lens to inform coding and theme development [22]. In this case study, we used technical capabilities identified by Forsgren et al. (2018) as the framework to identify the application of the continuous delivery principle. These principles include 1.) Version Control, 2.) Deployment Automation, 3.) Continuous Integration, 4.) Trunk-Based Development, 5.) Test Automation, 6.) Test Data Management, 7.) Shift Left on Security, 8.) Loosely Coupled Architecture, 9.) Empowering Teams to Choose Tools, 10.) Monitoring, 11.) Proactive Notifications.

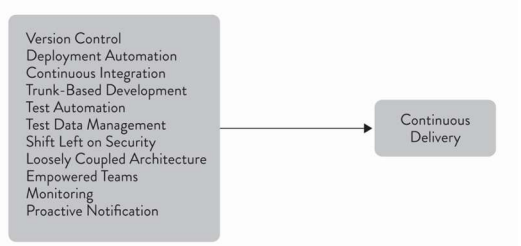

Figure 1 Technical capabilities that enables continuous delivery.

The results of previous research [5] prove that applying the aforementioned technical capabilities has a considerable influence on realizing the principle of continuous delivery. Further research [13] found that applying the continuous delivery principle had a positive impact on the company in terms of organizational performance and psychological impact on employees. We used thematic analysis to prove this impact.

Journal of Theoretical and Applied Information Technology
30th November 2022. Vol.100. No 22

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

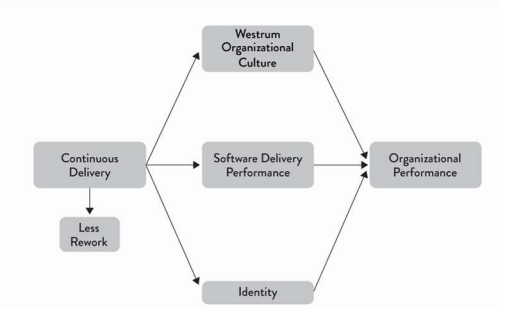

Figure 2 Continuous delivery impact on an organization.

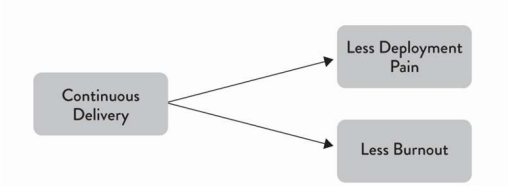

Figure 3 Continuous delivery impact on team's psychology.

We found three major themes from the data collected from the observations. The themes include the current implementation of continuous delivery, the initiatives or changes made, and the impact on the organization. We expected these themes to be able to answer the defined problem formulation.

3.3.2. Quantitative Analysis

All data provided by the company related to metric or quantitative data were analyzed first to determine whether the data was relevant to the research results. Quantitative data served as supporting data for the findings of the qualitative analysis that was carried out.

The company collects data from various platforms used in the development process (Asana, Gitlab). From the results of data exploration from tools or platforms used by the company, several metrics or data related to the development process, such as deployment frequency or the number of bug reports, were used in the study. These data were generated and retrieved directly from the platform based on the activities carried out by the company's development team on the platform used. Unfortunately, other data, especially on the Gitlab platform, look random and do not appear to have a specific pattern, so this case study did not use these data.

4. RESULTS

4.1. Analysis of Current Continuous Delivery Implementation

After collecting development data and conducting interviews with key employees from PT XYZ, we have uncovered several practices aligned with continuous delivery implemented by the engineering team. We discuss development practices implemented by PT XYZ based on related continuous delivery technical capabilities.

4.1.1. Architecture

PT XYZ develops and maintains a system that facilitates transactions for mutual fund investment. Price changes at the end of the working day for a mutual fund due to the complex administration process behind mutual fund valuation. This behavior makes mutual fund prices not fluctuate as often as stock. The transaction is also processed for a few days before users finally get the actual unit. This lesser fluctuation in price is why the company aims to promote mutual fund investment to the more general audience, even to people new to investing activities and the financial market.

This business process creates requirement which needs the engineering team to create a system that needs to be easily understood by people new to investing. When creating the system for the first time, the management team of PT XYZ did not know what kind of user interface would be accepted by ordinary people, primarily since investing apps were known to be complex at the time. Due to this minimal knowledge, the engineering team chooses to implement the most flexible architecture they can think of at the moment. Then they came up with the idea of creating a web app that would be attached to a mobile app through WebView.

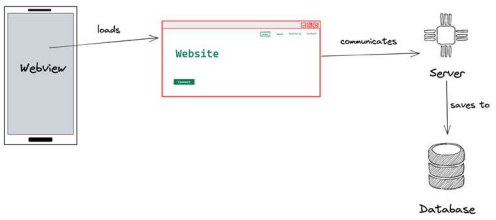

Figure 4 Initial PT XYZ architecture idea

During the creation of the system, mobile apps are massively growing. Therefore, the management team decided that the app should be released on iOS or Android through each official application store. However, releasing an app version in each store requires review, adding additional hours to one workday. This additional time sometimes slows PT XYZ from delivering bug fixes, mainly if fatal ones are found in production.

Due to these circumstances, PT XYZ decides to create most application views inside a WebView. WebView is a mobile development concept that attaches a browser view to mobile apps. This browser will load the URL of the web app. Since the application view is mainly inside the web, PT XYZ

can update the application view without waiting for each application store review.

The structure of the mobile application using WebView enabled the development team from PT XYZ to meet the company's needs to change application content quickly. According to the CTO of PT XYZ, WebView enables PT XYZ to improve the application quality without waiting for the review process from the application distribution center of each mobile platform. In addition, this structure allows PT XYZ to update the application without uploading a new version to each platform's distribution center, thus eliminating submission review time.

"WebView helps us to release changes more quickly, since most of our users come from mobile apps. We can rollout features faster without having to wait for AppStore or Play Store review process."

Chief Technology Officer of PT XYZ

Using WebView architecture also creates challenges for the development team. For example, the web team said that they do not find many examples of applications using WebView as the primary application architecture. This situation made the team create custom solutions every time a problem happens, and the team usually has difficulty finding solutions from other programming forums.

"We create our own solution to manage communication between our web app and native app. We have to create a custom solution since we do not find solutions elsewhere and WebView – native app communication seems uncommon in any other application."

Tech Lead from PT XYZ

The web and server-side teams still structure their application in the monolith for how the codebase is structured. This structure has been the same since PT XYZ began creating the app. For every feature added inside the application, the teams rely on folder structuring to give a sense of separation. However, one of the company's engineering leads expresses concern that this monolith architecture has reached its limit. This concern is because the pipeline process has become slower as the codebase grows.

"Over time, I saw increasing build duration and server spec needed to run the build process. I think the project has become too big for a build process."

Tech Lead from PT XYZ

4.1.2. Version Control

PT XYZ implements version control on their application code. For saving application code, PT XYZ uses a third-party service called Gitlab, which uses git as the primary version control technology. Every development team in PT XYZ follows different naming conventions and practices different branching strategies. The similarity in using version control lies in how every development team in PT XYZ marks its release. They mark every version ready to release using the tag feature in git.

Version control has become an essential tool for PT XYZ developers. Version control allows them to revisit previous versions if they remember which version to target. In addition, version control helps PT XYZ to study the previous implementation, revert code when needed, and track who does the critical change.

For application configuration in the web app, since the configuration is mainly configured in application code, saving application configuration is not much different from saving code into the repository. Every option for building a web app bundle is saved inside the repository in the form of application code or saved as a script inside pipeline code. The only configuration option not saved inside the repository is the environment variable. Environment variables are specific variables that can be different for each environment target. These variables are sometimes confidential, like a secret key controlling a specific service. Due to variety and secrecy, environment variables for web apps are not saved inside the repository. Instead, these variables are stored inside the pipeline service provided by Gitlab and not versioned. A file containing example values of environment variables stored in the repository for educating new developers.

Application and system configuration are saved inside the repository through Terraform files for server-side systems. These Terraform files store how the backend system is arranged and serves as a template for deploying new service. However, not all developers have access to every service inside the cloud provider, and not everyone understands how to maintain Terraform files. In PT XYZ, this responsibility to maintain Terraform files is moved to a different team, called the infrastructure team. Even if a separate team maintains these files, the server-side team still contributes and can easily access Terraform files.

4.1.3. Deployment Automation

When PT XYZ starts to build the application, all deployment processes are done manually using scripts triggered by a developer from their work laptop. There are four crucial steps of the deployment:

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

1. Build the web app bundle
2. Upload build artifact into the server
3. Invalidate cache server
4. Upload build artifact

Along the way, PT XYZ improved its deployment process by utilizing an automated pipeline feature from its third-party provider. The version control provider also provides an automated pipeline service that utilizes the version control features, as seen in Figure 5. In this flow, every git tag created from a specific version control commit will form a set of processes called a pipeline. This pipeline consists of jobs that will do the deployment process described above, with the new addition of informing all people through communication channels if the deployment succeeds. If a developer triggers this pipeline manually using a specific button, the jobs will start until the deployment process is finished. With this flow, the four deployment steps above are being run by the pipeline, not executed manually. The automated flow ensures no human error while deploying the app.

This deployment flow leaves numerous manual triggers. The first one is the process of creating a git tag. This process is manual so the tech lead can decide the best version to be tested and deployed to production. The second is the process when developers decide to allow the pipeline to run. This process allows developers to give the product and QA team some time to validate and decide whether the app should be released. Besides these processes, all other processes in this deployment flow are run automatically.

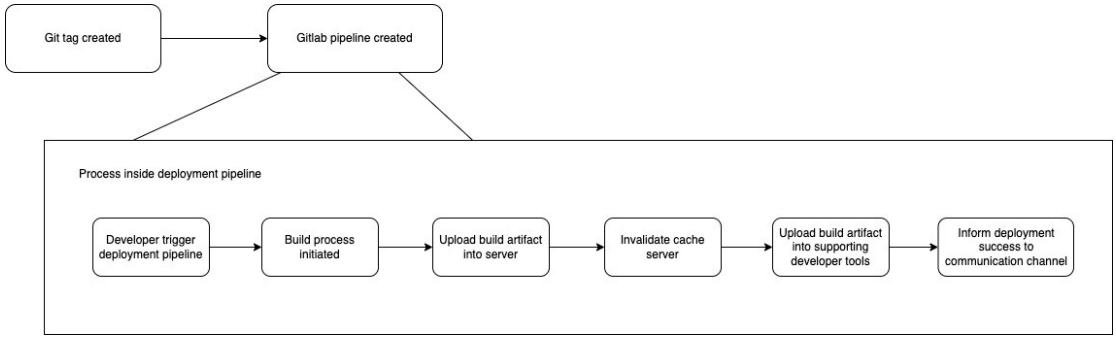

Figure 5 Deployment flow of PT XYZ web platform, which utilizes automation

4.1.4. Continuous Integration

PT XYZ runs an automated code test every time developers submit their commit into version control. In addition, PT XYZ has prepared several checks for every commit submitted, from automated code guideline tests, unit tests, and coverage reporting. This process allows developers to receive automated feedback when submitting their change proposals.

However, PT XYZ does not run the build process automatically on all commits for the build process. The build process only runs for commits tagged with a version number. This decision is made because the unit test can be run using source code without requiring the app to be bundled. This condition is normal in JavaScript apps. Before running the unit test, adding a build process is considered as adding extra time to the whole testing pipeline.

For features that require more detailed and qualitative test by humans, developers in PT XYZ communicates this need through an office communication channel such as Slack. Even if the web app build process is not designed to run automatically on all commits, developers can deploy every commit in version control into a new isolated environment. This capability allows test execution without interfering with irrelevant features. The new environment creation still needs to be done manually, either through the developers' laptop or through a form submitted on the Slack channel. This will trigger the bot to create a new test environment based on the submitted form.

After a new environment is created, developers usually announce the new testing environment in the company's testing forum. Several members from the QA team will check, and if passed, the changes will be processed further, either merged to the main branch or released to production if the change is already in the main branch. If the changes do not pass the test, the engineering team will continue to fix them until passed. Unfortunately, we cannot measure how long the engineering team gets the testing feedback and fixes the broken build due to lack of record. According to the interview with PT XYZ tech lead, the time to receive feedback may varied, but testing for production release usually takes about three to four hours.

"Before releasing features or fixes to users, we give our apps to tester using controlled environment. […] Testing time varies depending on how big the feature

or the fix is. We do not measure precisely how long it takes for us to get feedback. Based on experience, we usually get feedback in about 3-4 hours. Big features can take days for us to receive feedback from testers."

Tech Lead from PT XYZ

4.1.5. Test Automation

In PT XYZ, test codes are maintained by two teams. Each platform team will maintain the test for unit tests and tests that depend on application code. In this case, the web team will maintain the unit test for the web platform, and for server-side apps, the team handling server-side codes and database will maintain the unit test for this part. For more integrated tests, which involve many parts or features of the application, these integrated tests are handled by a different team, called the QA engineer team. While QA engineers can be considered developers, we saw that QA engineers rarely interact with the web team.

As explained in the continuous integration part, every commit submitted into the repository triggers test suites. This flow allows faster feedback for developers when they push commits into the repository. In addition, when tests fail, developers will try to fix the errors as soon as possible.

PT XYZ does not measure the number of bugs found in every testing phase. This condition makes it difficult to know the proportion of bugs found over time. PT XYZ also does not measure how long the engineering team takes to fix the error found during testing, and no measure compares the trustworthiness level of automated tests.

4.1.6. Test Data Management

PT XYZ engineering team has prepared a specific environment for testing. The environment created for testing is usually connected with a dataset separated from the production database but follows the same schema or data structure as in production. This separation ensures that the actual environment users are experiencing is not affected by the QA team's test scenarios. The data inside this particular dataset is usually random data inputted by testers.

The engineering team usually helps prepare the data based on the bug report analysis for scenarios that need specific data types. Unfortunately, there is no automated way to generate a dataset for tests. The engineering team must prepare and insert the data by themselves. Even if the production and test datasets were separated, the engineering team could not create additional test datasets, which means PT XYZ only has two types of datasets. This condition can create conflict if the test dataset needs to be used by two different scenarios and the data structure needed conflicts with each other. Luckily, according to the interview, this has not happened. There is also no report or observation regarding automated tests, which depends specifically on any structure of the test dataset.

4.1.7. Empowering Team to Choose Tools

PT XYZ empowers each team to explore every possible tool needed to create the best developer experience. Even several teams in PT XYZ allocate one specific day to ease feature development and app release, encouraging the team to explore new ways for them to develop tools or looking for new ways to develop and maintain the existing app in PT XYZ. Whenever people find something new, they are encouraged to share it with other team members through several forums, such as weekly sharing sessions or writing the findings in its tech blog.

This policy has allowed several development teams to improve their process. PT XYZ web team has created a new tool to help them collect release information by utilizing the allocated free time. This tool impacts the team deployment frequency, which will be discussed later.

Based on the data obtained through the interview process, several things can be improved from PT XYZ's policy. According to the technical lead web from PT XYZ, this policy has not yet succeeded in promoting innovation equally to every IT and development team member. New tool innovation mostly comes from a small number of senior-level engineers, and most engineers at a more junior level do not yet have the initiative or the idea to create new tools. There needs to be an effort to encourage knowledge sharing so that innovative senior technicians can share their knowledge so that other technicians are expected to follow suit.

"Current company policies have not been able to encourage innovation evenly. Until now, innovations related to new tools still come from more senior developers. Apart from that, there are rarely innovations. The average is still just doing the work at hand. […] There must be a sharing session to encourage more equitable innovation. When exploring, tech leads also need to follow up on whether the technology other engineers have explored can be useful."

Tech Lead from PT XYZ

4.1.8. Monitoring

For managing and monitoring tasks carried out by developers, the team at PT XYZ applies several processes from the Scrum principle, which are considered suitable for the situation at hand and the company's work culture. The process applied to each

platform team at PT XYZ is not always the same because each team is freed by the management of PT XYZ to adjust their process based on the situation at hand. Some of the ways that are applied to monitor the process of project/task work are as follows:

1. Weekly meetings.
   This activity is a weekly activity held by each team. It can be in the form of a report on the results of changes made during project work in the past week, and after that, there is usually further planning about what will be continued in the next week.
2. Daily stand-up.
   This activity is a daily activity to report the progress of each member of the development team. Each developer must convey what was done the previous day, today's work plan, and the obstacles to the current work.

PT XYZ has several tools to monitor applications released and servers running regarding the managed information system. These tools are divided into several categories:

1. Error monitoring (Sentry)
2. Performance monitoring (Sentry, AWS)
3. Monitoring logs (AWS)
4. User activity monitoring dan business metrics (MoEngage)

According to the product manager of PT XYZ, the current monitoring system is quite helpful in making business decisions. Monitoring related to user activity is enough to help product and business teams to measure application achievements from a business perspective. The metrics observed on this platform are often used for consideration of improvements and feature additions made.

"The user activity monitoring system currently installed is quite helpful for us in making business decisions. For example, changes to the number of daily active users will make us take different decisions regarding future feature developments."

Product Manager of PT XYZ

4.1.9. Proactive Notification

PT XYZ also implements a notification system for important system events for ease of monitoring. These events can be an event that occurs at a specific time, such as a service that is not indicated to be active for a while, or it can be in the form of reaching a specific size limitation, such as the response of service suddenly running slowly, or the server capacity is not sufficient. These notifications are sent in an email and through work messaging applications such as Slack. In PT XYZ, the notifications which the engineering team watched frequently are:

1. Runtime errors caught by Sentry, especially if the stack comes from source code
2. Deployment process notifications
3. Error notification on query execution

Some engineering team members admit that the error notification sometimes still catches irrelevant errors outside the source code for runtime error notifications. This condition makes several engineering team members ignore notifications.

4.1.10. Trunk-based Development

While doing development, PT XYZ still implements the feature branching model. This branching model means developers will create a new branch for every new feature or bug fix. The work can take days to several weeks, which is the average age of merge requests in PT XYZ. Some merge requests are open for two weeks or more. When the work is done, and the changes have passed several tests, the set of changes inside the branch will be merged into the main branch.

The downside of this model is that PT XYZ will create a lot of branches each day, and these branches have a chance of not being merged on that day due to requirement changes or simply due to merge conflicts. Branches not merged into the main branch for too long will cause a specific branch to diverge further, making it harder to merge to the main branch and increasing the chance for merge conflict. This situation possibly creates a cycle of delayed merge requests to the main branch.

This downside can be seen during observation. When releasing new features, specifically big ones, PT XYZ usually spends one to several days rechecking regressions and stabilizing their code due to merge conflicts. Integration phases can still be seen during development.

4.1.11. Shift Left on Security

For PT XYZ, the focus on security is relatively new. During the observations made, PT XYZ is practically just starting to integrate the security engineer team into the feature development process. During the observation, the features that were developed and passed the security inspection were only several features related to user data, such as login, registration, and user information page, as well as security-related features such as pin input and login activity tracking. Security inspections on features have not been carried out every time the feature is developed because the development team and product team from PT XYZ have not been able to estimate how long it will take to add security inspections for each feature designed and developed. For features that have been checked for security, there is no exact

measurement of how long the security inspection process takes.

PT XYZ is also starting to involve the security engineer team in the company's software development phases. The security engineer team is involved when the feature design is done. When designing features, the security engineer team generally provides comments regarding the flow of features that have been designed. If the feature design is deemed to be following security standards, the security engineer team will approve the feature. Involvement in feature design is currently not carried out for all features and is limited to features that are considered to be related to users' personal data or financial data.

The security engineer team is also involved in the code development process by creating automation to perform static code security checks using a specific tool called *Horusec*. According to the Android tech lead, *Horusec*'s recommendations help provide an early security review of the app, so it feels like the security engineer team is involved in the development process. During the observation, it was also seen that the security engineer team also provided reviews for Android application code related to application security. The same is true for iOS apps.

"I feel the involvement of PT XYZ's security engineer team in securing Android applications. The security engineer team sometimes presents security findings related to the application to us, which we then make improvements, and the code is reviewed together. The security engineer team also uses code checking tools to automatically review Android code according to the security standards they set."

Tech Lead Android PT XYZ

Regarding the testing process and application release, the security engineer team is involved in carrying out final checks on the features that are deemed necessary for security checks. During the observation, an idea emerged from one of PT XYZ's tech leads to conduct regular penetration testing. The idea of penetration testing is still in discussion with management, so it has not been carried out during observation, but it can be a sign that the security engineer team will be increasingly involved in the development process.

4.2. Changes in Development Process During Observation Period

During the observation period of PT XYZ, several initiatives are being rolled out by either the management team or members of the engineering team. We discuss these initiatives and relate these to continuous delivery principles.

4.2.1. Improvement in Incident Reporting and Handling

As the tech leads and management of PT XYZ learn about continuous delivery principles and their experiences in handling critical incidents, they see the need to track all the reported incidents. The company needs to track the incident's timeline, the source of the report, how the team handles the incident, and what the team has learned. This situation inspires the engineering team to improve how incidents are handled in the company.

The new flow of incident handling involves an exclusive communication channel used for emergencies due to incidents. At first, the whole engineering team and the product team agreed on the incident definition. The team agreed that an event could be categorized as an incident only if the application was not usable. If the application happens to be down for a moment, anyone who is informed should alert the engineering team through this specific channel. One person will record the incident's timeline, and one person from the engineering team will lead the fix. After the incident is resolved, the engineering team will gather and conclude the incident report with what they have learned to resolve the incident.

The incident reporting has helped PT XYZ monitor how PT XYZ's engineering team handles incidents. The engineering team can now track how long the downtime occurs, whether a new build or another team causes the incident, and the change fail rate. Both the product team and engineering team hope this can improve the duration of how long critical incidents are resolved.

4.2.2. Automated Changelog Generation

Since the pandemic occurs, PT XYZ has issued a *Work from Anywhere* (WFA) policy. This policy allows the engineering team to work at home or in the office, anywhere they want, with the condition that the team should be available during the working hour. As the impact of this policy, most PT XYZ engineering teams are not collocated in the same working space. This situation creates a delay in communication.

The delay in communication sometimes affects deployment. In PT XYZ, the engineering team must communicate to the product team and to the C-levels which changes are released. Sometimes it takes 5-10 minutes for the maintainer of the repository to wait for engineering team feedback. As the engineering team grows, these minutes have grown into an hour or more delay just to ask what changes are implemented. This delay is seen as unnecessary, and one of the

engineering team members has an idea to generate the changelog.

Rather than asking each team member manually through a communication channel, one of the team members suggests collecting commits from the main branch and treating it as a changelog. The commit format should be machine-readable to be read by script and automated. Once the script has been made, the script's creator then introduces this script to the engineering team. The whole team accepts this tool with an open mind, which has affected the team's deployment frequency.

4.3. Continuous Delivery Impact on Organization

During the observation, we found several impacts on the organization related to implementing the key continuous delivery capabilities. This section describes the impact on team performance and the psychological impact on the IT team of PT XYZ.

4.3.1. Impact on IT Team Performance

Figure 6 shows that the deployment frequency from August 2021 onward is more frequent on average. This improvement is due to the changes in the web development team process. At the end of August 2021, the web team used the automated changelog generation tool to help the team deal with merge requests. The company executive and the existing team members support this new tool development. This empowerment creates new momentum for the team to release more frequently.

The increase in deployment frequency also creates a sense of relief for the product team since increased deployment frequency means more chances for the product team to update the application. From the opinion of PT XYZ's product manager, this allows the product team to adapt more quickly whenever changes, either fixes or new features, are needed to be released sooner. Whenever a release goes wrong, the product team can also revert it to any tracked version since the application code is tracked using the version control system.

Unfortunately, the increasing number of bugs follows the increase in deployment frequency in the web team. Although PT XYZ has the policy to halt the deployment process if the build has critical bugs, this does not decrease the number of bugs due to most bug reports on the web being categorized as low impact. The bug report categorized as low impact is usually cosmetic errors, such as misplaced elements or wrong copywriting. This bug is usually covered by redeploying the correct version, supported by increasing deployment frequency capability. The number of critical bugs is decreasing on average due to the policy of not releasing an app when a critical bug is found while testing.

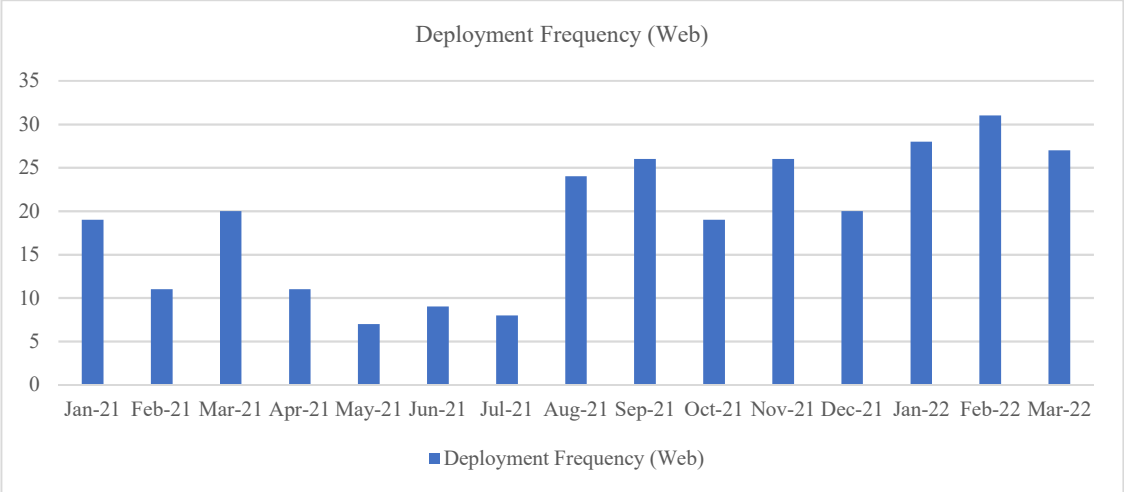

Figure 6 Web team deployment frequency

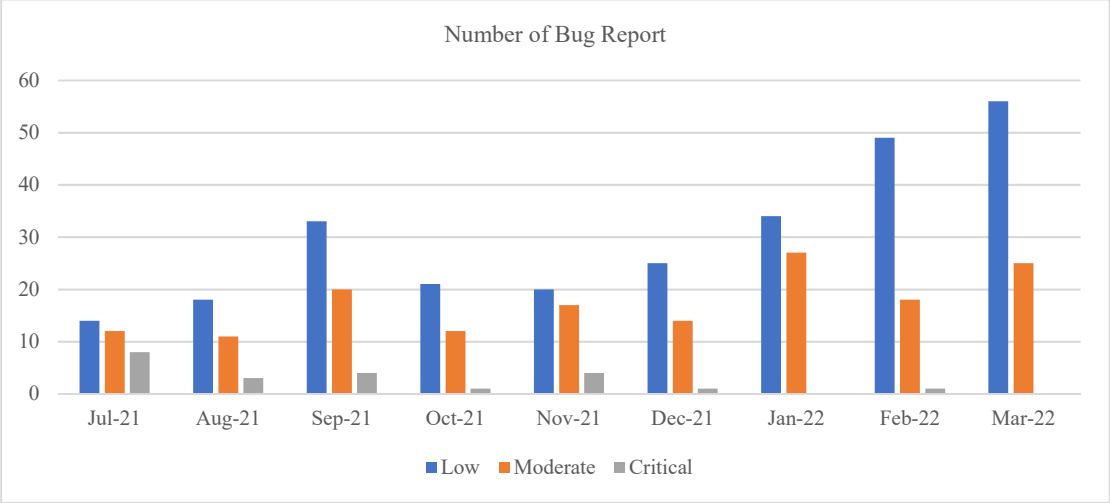

Figure 7 Number of bug reports related to PT XYZ web team

4.3.2. Impact on Team's Psychology

Several key capabilities implemented according to continuous delivery has brought positive reaction to the company's management and engineering team. According to the interview with the CTO, WebView architecture is considered to help the development team to release important fixes faster, especially for mobile applications that must go through a review from the application distribution center of each platform. The executives expressed satisfaction with the faster release rate, indicating stakeholder satisfaction with the performance of the PT XYZ development team.

"In my opinion, the WebView architecture that is applied is sufficient to reflect the application of the continuous delivery principle. This architecture allows us to carry out the delivery process to users quickly and continuously. So far, there have been no complaints regarding delays in the release of fixes or features."

CTO of PT XYZ

Apart from architecture, other key continuous delivery capability points such as the use of version control and a pipeline system to automate the build, test, and release processes are also quite helpful by many members of the development team at PT XYZ. The implementation of version control allows the product team and IT team of PT XYZ to restore the system to the previous correct version if an error occurs. The ability to revert to a previously correct release version provides a sense of security for developers and product teams when releasing a feature with substantial changes. Then the company's policy of giving more free time to the exploration process has inspired the emergence of various tools that assist development, such as the automatic changelog generator tool described in the previous section.

"The use of version control for application code is quite a relief for us because if there is a fatal error in development, we can easily return to the previous correct version."

Senior Web Developer PT XYZ

5. DISCUSSIONS

5.1. Discussion on PT XYZ Continuous Delivery Implementation

According to Forsgren et al. (2018), high-performing teams tend to agree on architectures with the following characteristics: 1.) Testable without the need for an integrated environment. 2.) Can carry out the deployment or release process without relying on other applications or services.

The architecture applied by PT XYZ can be considered to meet these characteristics. Using WebView causes PT XYZ only to need to test and release the web application if there is a change in appearance without waiting to update the mobile application. Changes in appearance are among the changes that occur most frequently in applications, so the choice of this architecture is one of the keys that cause PT XYZ to release various changes quickly.

Apart from architecture, other key continuous delivery capabilities such as version control, deployment automation, and continuous integration are also seen being implemented by PT XYZ. The development team of PT XYZ can answer interview

points related to this capability completely. Even the development team of PT XYZ also created a new initiative that was built on one of the key capabilities of continuous delivery. This proves that continuous delivery capabilities have been implemented.

Besides the technical capabilities mentioned above, researchers found various technical capabilities whose implementation is still less visible. For example, testing-related capabilities such as continuous testing and test data management do not yet have sufficient evidence. Although the shift left on security capability has been implemented, the measurement of this capability is still minimal. Monitoring capabilities in terms of infrastructure and applications and proactive notification capabilities have also not significantly impacted the organization.

5.2. Challenges of Continuous Delivery Capabilities

During PT XYZ's journey to implement continuous delivery, the company faced several challenges. The challenges faced can be categorized into internal or external to the company.

The first challenge comes from outside of the company. This challenge is the difficulty of ensuring that the critical capabilities of the continuous delivery principle are correctly implemented. The difficulty in ensuring the correct application of the continuous delivery principle is due to the continuous delivery principle and the DevOps movement, which do not yet have a formal application framework [15]. Furthermore, continuous delivery and the DevOps movement were developing among professionals [1] and interpreted differently by various practitioners discussing it. It may also explain why it is difficult to define a solid framework for assessing the implementation of continuous delivery.

Another challenge comes internally, namely how the development team of PT XYZ proves the impact of the continuous delivery principle on organizational performance. The data collection results show that PT XYZ has not done many measurements related to implementing key continuous delivery capabilities. The lack of data makes it difficult for PT XYZ to measure the effectiveness of the key continuous delivery capabilities implemented. The problem in the first challenge regarding the absence of a solid framework for implementing continuous delivery may also be the cause that contributes to this second problem, considering that to take measurements, a team needs to know what it wants to measure.

To overcome this, PT XYZ can start looking for a framework or points for applying the continuous delivery principle that can be used as a benchmark. The key capability of continuous delivery (Forsgren et al., 2018) and the three basic principles of DevOps (Kim et al., 2016) can be the initial framework for PT XYZ to adopt the principle of continuous delivery.

The last internal challenge is how PT XYZ balances the capabilities of the applied continuous delivery principle. As shown in Figure 6, PT XYZ feels an increase in the number of deployment frequencies, which is the impact of applying several principles of good continuous delivery capabilities. However, this is also accompanied by an increase in low-level bug reports, as shown in Figure 7. Furthermore, observation and interview results show that technical capability related to testing is less visible in PT XYZ. This situation can be the reason for what happened in Figure 7, even if Figure 6 shows positive results due to other technical capabilities, such as architecture, version control, and deployment automation implemented quite well.

5.3. Impact of Continuous Delivery Implementation on PT XYZ

When viewed from the interview results, the first thing that can be seen regarding the impact of the application of the continuous delivery principle on the company is the psychological impact it brings to the IT team and developer of PT XYZ. Top-level management, product team, tech leads, and developers from PT XYZ responded positively to applying various capabilities of the continuous delivery principle. These responses align with the finding that continuous delivery can reduce employee burnout [5], [13].

The reduction in deployment pain is also evidenced by new tools that make it easier for teams to release applications. This new tool can be realized because of the policy of PT XYZ that encourages the exploration of new tools for development and the implementation of version control. Statistically, an increase in deployment frequency also reduces deployment pain. This fact is related to the consideration that the team will deploy more often because the deployment process becomes easier after new tools that help the deployment process is implemented correctly. This increase in deployment frequency also indirectly affects organizational performance because the development team can quickly adapt fixes to the application.

Apart from increasing deployment frequency, few other positive impacts have been found from the principle of continuous delivery. This situation happens because the company has not fully implemented the critical capabilities of the continuous delivery principle properly. New problems also accompany the increase in deployment frequency, for example, an increase in the number of bugs with low

impact. However, the application of the continuous delivery principle, although not yet fully implemented, can positively impact the organization. This situation can be an impetus for the development team to continue implementing the key continuous delivery capabilities, hoping that a better and more complete implementation can bring a much more significant and positive impact to the organization.

5.4. Conclusion and Recommendation for Future Research

This study explains how a company runs a software development process following the principle of continuous delivery and uncovers the challenges and impacts of applying the principle of continuous delivery to a company. The conclusions obtained from the research include the followings.

PT XYZ applies several points of continuous delivery technical capability well. PT XYZ consciously chooses the right architecture to support the needs of rapid release and uses version control that supports the implementation of deployment automation and continuous integration capabilities. PT XYZ also implements other capabilities such as monitoring and empowering teams to choose tools. However, PT XYZ has just started to shift left on security and needs to be supported so the organization can feel the benefits. In addition, other continuous delivery capabilities such as trunk-based development, test automation, test data management, and proactive monitoring have not been adequately implemented, and no clear evidence or measurements have been found for these capabilities.

PT XYZ has various challenges in implementing the continuous delivery principle, both internally and externally. Internally, the lack of measurement to prove it, as well as the challenge of balancing the impact of implementing continuous delivery capabilities, are seen to be faced by the company. In addition, companies need to balance implementing better deployment automation capabilities with implementing testing-related capabilities such as test automation and test data management because the number of bugs found is increasing despite improving deployment frequency. Externally, practitioners promoting the DevOps movement and continuous delivery have not agreed on a solid framework to validate the continuous delivery application. This situation caused difficulties for PT XYZ to determine whether the development process has followed the principle of continuous delivery.

Continuous delivery positively impacts PT XYZ's software development process, even though PT XYZ has not fully implemented the continuous delivery principle properly. PT XYZ feels the positive impact on performance through increasing deployment frequency. The increase in deployment frequency allows PT XYZ's development team to ship features and bug fixes more quickly, thus affecting user and management satisfaction. PT XYZ also feels a positive psychological impact from reduced deployment pain and employee burnout.

Judging from the continuous delivery capability that PT XYZ carried out during the study, we suggest continuing the application of the continuous delivery principle capability, along with some improvements. With these findings, PT XYZ needs to balance the increase in deployment frequency with increased test coverage and measurement of test reliability. Automated testing has not covered a large part of PT XYZ's project, and the level of reliability of the automated test series at PT XYZ has not explicitly been observed and controlled. PT XYZ also needs to improve the process of monitoring and measuring its development process so that the development team can better apply the principle of continuous delivery and have more impact in encouraging the organization to achieve its goals.

For further research, we suggest further researching each technical capability related to the principle of continuous delivery and its impact on team performance. The study results show that applying some of the critical technical capabilities of the continuous delivery principle can increase the performance of the company's IT team. Future research can highlight how each technical capability affects the performance of a company's IT team based on specific measurements. Further research into team culture will also help to uncover the psychological benefits of applying the essential capabilities of continuous delivery.

REFERENCES:

[1] J. Davis and R. Daniels, *Effective DevOps: Building a Culture of Collaboration, Affinity, and Tooling at Scale*. O'Reilly Media, 2016. [Online]. Available: https://books.google.co.id/books?id=nO1FDAAAQBAJ

[2] P. Swartout, *Continuous delivery and DevOps: a quickstart guide*. Packt Publishing Birmingham, 2012.

[3] Fowler M, "Continuous Integration," May 01, 2006. https://www.martinfowler.com/articles/continuousIntegration.html (accessed Oct. 20, 2021).

[4] J. Humble and D. Farley, *Continuous Delivery: Reliable Software Releases through*

Build, Test, and Deployment Automation. Pearson Education, 2010. [Online]. Available: https://books.google.co.id/books?id=6ADDuzere-YC

[5] N. Forsgren and J. Humble, "The Role of Continuous Delivery in IT and Organizational Performance," *SSRN Electronic Journal*, Oct. 2015, doi: 10.2139/SSRN.2681909.

[6] Z. Dragičević, S. B.-I. S. Conference, and undefined 2020, "Agile Development Process in The Software Factory of the Future," *sm.ef.uns.ac.rs*, 2020, doi: 10.46541/978-86-7233-386-2_43.

[7] D. G. Feitelson, E. F. Facebook, and K. L. B. Facebook, "Development and Deployment at Facebook," 2016.

[8] C. W. Fuller, "Continuous Integration/Continuous Delivery Pipeline for Air Force Distributed Common Ground System," Air Force Institute Of Technology, 2020.

[9] M. Rubert and K. Farias, "On the effects of continuous delivery on code quality: A case study in industry," *Computer Standards & Interfaces*, vol. 81, p. 103588, Apr. 2022, doi: 10.1016/J.CSI.2021.103588.

[10] M. Sigfast and F. Olsson, "Benefits of continuous delivery for Sigma IT Consulting." 2018.

[11] Puppet Labs, "2015 State of DevOps Report," *Puppetlabs*, no. 877, 2015.

[12] A. Hemon-Hildgen, F. Rowe, and L. Monnier-Senicourt, "Orchestrating automation and sharing in DevOps teams: a revelatory case of job satisfaction factors, risk and work conditions," *Article in European Journal of Information Systems*, 2020, doi: 10.1080/0960085X.2020.1782276.

[13] N. Forsgren, J. Humble, and G. Kim, *Accelerate: The Science of Lean Software and DevOps: Building and Scaling High Performing Technology Organizations*. IT Revolution Press, 2018. [Online]. Available: https://books.google.co.id/books?id=Kax-DwAAQBAJ

[14] Puppet Labs, "2021 State of DevOps Report," *Puppetlabs*, 2021.

[15] Rütz and Martin, "Devops: A Systematic Literature Review," 2019.

[16] M. McCormick, "Waterfall vs. Agile methodology," *MPCS, N/A*, 2012.

[17] K. Beck, M. Beedle, A. van Bennekum, and A. Cockburn, "The agile manifesto," 2001, Accessed: Oct. 20, 2021. [Online]. Available: https://courses.cs.ut.ee/MTAT.03.094/2015_fall/uploads/Main/SE2014-handout11.pdf

[18] L. Williams and A. Cockburn, "Agile software development: It's about feedback and change," *Computer*, vol. 36, no. 6. 2003. doi: 10.1109/MC.2003.1204373.

[19] C. Ebert, G. Gallardo, J. Hernantes, and N. Serrano, "DevOps," *IEEE Software*, vol. 33, no. 3, pp. 94–100, May 2016, doi: 10.1109/MS.2016.68.

[20] G. Kim, J. Humble, P. Debois, and J. Willis, *The DevOps Handbook: How to Create World-Class Agility, Reliability, and Security in Technology Organizations*. IT Revolution Press, 2016. [Online]. Available: https://books.google.co.id/books?id=ui8hDgAAQBAJ

[21] V. Braun and V. Clarke, *Thematic analysis.*, vol. 2. 2012. doi: 10.1037/13620-004.

[22] V. Clarke, V. Braun, and N. Hayfield, "Thematic analysis," *Qualitative psychology: A practical guide to research methods*, vol. 222, no. 2015, p. 248, 2015.