An investigation into the effectiveness and impact of a Continuous Delivery pipeline upon university-level games development teams?

Frost Donovan

***Abstract*—What's the problem? What am I looking at? How does that help solve the problem?**

Opening, Challenge, Action, Resolution

Continuous Delivery (CD) is a technique designed to increase reliability and consistency with delivery of builds. This can then help to increase frequency of testing, the accountability of the development team, as well as the teams transpareny and *"all being on the same page ness"*. This can increase team moral as well as stakeholder confidence, as both are able to regularly see the current state and rate of progress for the project. This encourages regular analysis of development pace and project scope, both internal and external.
This stakeholder confidence and awareness of scope is a common problem within student development teams, so this investigation will answer to what extent the deployment of a CD pipeline will help to reduce these problems.

I. Introduction

A. What is Continuous Delivery?

CD is an expansion of Continuous Integration (CI), a pipeline for the continuous integration of code, being developed on seperate branches, into the main branch. This code is merged into the main branch and then tested to ensure there are no merge conflicts or obvious errors thrown from the combined code. Assuming all of these tests pass, this merged code is pushed to the project repository. A key part of this pipeline is that the entire process is automated, ensuring minimal time is wasted waiting for tests to run or code to be uploaded. [1], [2]
CD takes this one step further, in that after a project passes through the CI pipeline, the code is then compiled, packaged, and further tests that require the project to be compiled can be run. These tests should not be run before other tests that don't require compilation due to Fail Fast principles [3], [4]. Assuming there are no failures during compilation or testing, the built program can then be uploaded to somewhere where it can be easily accessed. Here, it is important to make a distinction between Continuous Delivery, where the build is available internally but not to customers, and Continuous Deployment, where the build is pushed straight out to the current product users. **WHICH ARE WE LOOKING AT? LIMITATIONS! WANNA DO DELIVERY THAT SHITS SICK WOULD NEED TO GET SET UP WITH BUTLER FOR ITCH, LOOKS EASY THO**

B. Benefits & Drawbacks of CD

What benefits is it supposed to have [1], [5]? Drawbacks [5]?

II. Background & Supporting Literature

Has this been done before in academic setting [6]–[9]? Links to other things - CI [2], Unit tests, regular product reviews, stakeholder (supervisor) confidence, git flow [10] Best practices [11]?

III. Research Question

From the above sources, I have formed **The actual question**

A. hypothesis & null hypothesis

IV. Artifact

A. What will be made

CD pipeline utilising Github Actions. Tool to set up secrets? Would be sick https://docs.github.com/en/rest/reference/actions#secrets
Autoupload to steam & itch! https://itch.io/docs/butler/

B. How will I ensure Quality

Quality control. Roadmap? Unit Testing? Integration testing?

C. How will I create it

D. Why will this answer the questions

V. Research Methodology

A. Experimental Design

B. Limitations

Time, resources

C. Sampling Plan

Sample size, sampling method

D. Data management plan

Managing, collecting, & storing data

E. Data Analysis

T-test?

F. Ethical Considerations

VI. Appendix

Data analysis code, supporting screenshots, list of unit tests & testing plan

References

[1] J. Humble and D. Farley, *Continuous delivery: reliable software releases through build, test, and deployment automation*. Pearson Education, 2010.

[2] S. Pittet, "Continuous integration vs. continuous delivery vs. continuous deployment."

[3] J. Shore, "Fail fast [software debugging]," *IEEE Software*, vol. 21, no. 5, pp. 21–25, 2004.

[4] Atlasian, "Bamboo best practice - using stages," 2021.

[5] L. Chen, "Continuous delivery: Huge benefits, but challenges too," *IEEE Software*, vol. 32, no. 2, pp. 50–54, 2015.

[6] S. Krusche and L. Alperowitz, "Introduction of continuous delivery in multi-customer project courses," in *Companion Proceedings of the 36th International Conference on Software Engineering*, ICSE Companion 2014, (New York, NY, USA), p. 335–343, Association for Computing Machinery, 2014.

[7] S. Krusche and B. Bruegge, "User feedback in mobile development," in *Proceedings of the 2nd International Workshop on Mobile Development Lifecycle*, MobileDeLi '14, (New York, NY, USA), p. 25–26, Association for Computing Machinery, 2014.

[8] B. Bruegge, S. Krusche, and M. Wagner, "Teaching tornado: From communication models to releases," in *Proceedings of the 8th Edition of the Educators' Symposium*, EduSymp '12, (New York, NY, USA), p. 5–12, Association for Computing Machinery, 2012.

[9] K. Kuusinen and S. Albertsen, "Industry-academy collaboration in teaching devops and continuous delivery to software engineering students: Towards improved industrial relevance in higher education," in *2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering Education and Training (ICSE-SEET)*, pp. 23–27, 2019.

[10] V. Driessen, "A successful git branching model," 2012.

[11] P. M. Duvall, S. Matyas, and A. Glover, *Continuous integration: improving software quality and reducing risk*. Pearson Education, 2007.