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

Frost Donovan

***Abstract*—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 separate 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 users, and Continuous Deployment, where the build is pushed straight out to the current product users. core to agile methodology [5]

B. Benefits & Drawbacks of CD

The primary benefit of CD is reduced cycle time - a reduction in the time it takes for a change in the project to happen and then for the user to have that change applied to their version of the software. This principle is core to the agile methodology, being the very first principle in the agile manifesto [5]. This faster cycle time means that feedback from active users can be obtained much quicker, both on the effectiveness of bug fixes and also on new features. Agile is designed to avoid the pitfalls of Waterfall [6], one of which is the commitment of significant time and/ or resources into features that either are unattainable, or not actually wanted by the user. While other agile methods, such as Scrum or Extreme Programming [7], [8], can be an important part of the agile process, the effectiveness of a development team in delivering *value* is always going to be dependant on the speed and reliability with which user feedback can be obtained.

A benefit of this fast cycle time is not only are players able to see these changes faster, but stakeholders and publishers are able to see development progress, both regularly and on demand. This can be a significant step to building trust between a development studio and publisher, especially if the studio is new or doesn't have an existing relationship with the publisher [9].

Part of the CD pipeline is testing, with a suite of unit tests being run on the code as part of the build process. As this build process is run consistently, rather than all at once leading up to a main release, this means bugs are found incrementally, stopping the accumulation of technical debt, reducing the cost to fix bugs, and reducing stress on programmers by preventing an overwhelming influx of bugs. While these unit tests will likely catch a lot of bugs, some bugs will only be caught during human playtesting. This decreased cycle time means that human playtesting can happen sooner & more regularly, and fixes are delivered to testers & players almost immediately, rather than having to wait for the next release window.

Another strength of a Continuous Delivery pipeline is that it is fully automated. This allows less developer time to be spent setting up build or test environments and manually going through the build process, and more time on actually creating the product. This can be a *significant* time save, with some large scale projects reportedly taking weeks to set up environments ready to produce a release build [1], [10]. This system also significantly reduces the chance that there are any errors caused by mistakes during the build process as this build-release pipeline will have had many iterations of the product pass through it, before a major release deadline. This increases the reliability and stability of new releases.

II. Background & Supporting Literature

With these benefits in mind it raises two questions; If a CD pipeline is this important and valuable, is it being taught to

new developers in further education? If it is, is it actually as effective in practice with student teams as it is in theory?

There is limited literature relating to Continuous Delivery being implemented in an academic context, and *no* literature that I could find of this being implemented in a game development context. Even upon reviewing a literature review on rapid releases [11] there were *no* references to this within a games development context. This lack of literature provides a problem when attempting to find evidence on the performance of CD pipelines, however does highlight a *need* for further literature and case studies on the subject. There is even a lack of literature relating to the effectiveness of CD pipelines within an industry setting.

Relating to the effect of CD pipeline deployment in an industry setting, a literature review by Mäntylä et al. [11] carried out an investigation into firefox's transition "from a TR [Traditional Release] model of one release a year to an RR [Rapid Release] model where new releases come every 6 weeks" [11, pg 2]. This paper concluded that, while there are are many benefits of rapid releases in literature, in this case study the transition from traditional release to a rapid release process "[has] not significantly impacted the product quality" [11, pg 40]. It is worth noting however, that this conclusion has been drawn from an interview with a single Mozilla Firefox QA engineer, as well as the test execution data from 06/2006 to 06/2012. While this data allows a quantitative analysis of the number of tests run or bugs found, it neglects the qualitative side of quality testing, the user's opinion of the software, usability, and perceived work being put into the software. As such, while this case study can give insight into the quantitative effects, it has insufficient evidence to state that the "quality" of the product has not changed [12].

Given this lack, the papers being reviewed will be based around CD implementation in the academic setting of software development, a parallel field, but one which notably consists much more heavily of code, rather than the more even mix of skills and disciplines that is present within a game development context. As such, these papers all have a lack of analysis on how developers other than programmers respond to a CD pipeline, another case where there is a clear need for further study and literature.

Has this been done before in academic setting [13]–[15]? Links to other things - CI [2], Unit tests, regular product reviews, stakeholder (supervisor) confidence, git flow [16] Best practices [17]?

III. Research Question

From the above sources, there is a clear need for research into the practical effects of a Continuous Delivery pipeline within game development and game development education. From this knowledge, I propose this initial investigation into the effects of a Continuous Delivery pipeline upon university-level games development teams, with a focus on the confidence of the team, as well as the confidence of the team's academic supervisor in their team. This is purposefully broad, with the aim of encouraging and supporting further research into the topic. Given this, the following hypothesises will be investigated.

1) The use of a CD pipeline will increase a supervisors confidence in their team's ability to deliver a new, working build each week. (Q.S1)
2) The team's confidence and the supervisors confidence in being able to achieve everything within the scope of the project will be much more closely related with the use of a CD pipeline.(Q.S2,St1 plotted against time)
3) A CD pipeline will help developers understand the current state of the project (Q.St2)
4) A CD pipeline will help developers to always know why the work they are doing is being done (Q.St3)
5) A team using a CD pipeline will refine their scope more often (Q.St4)
6) A team using a CD pipeline will do more playtesting

IV. Research Methodology

A. Experimental Design

Environmental variation will be accounted for as all student team's within the study will be taken from the same population, Falmouth University Game's Academy. There may, however, be a relevant strata within this population that is not being accounted for, and that is the academic year the development team is in.

Environmental concerns will be accounted for as all students are taken from the same department at same university so are part of same strata. variation in sub-strata (year group)? This is being ignored due to limitations - there are only 15 team's per year so getting a sample size of 30 in a single year is not possible. Fully randomised design one hypothesis maps to one question on survey, apart from h2 which maps to two questions

B. Limitations

The primary limitation in this study is time. If this study were to be expanded, I would ideally like to follow team's from their first year all the way through to graduation, introducing half of the teams to the CD pipeline immediately and then tracking their results, feedback, and attitudes across all three years against the control groups. This could also be done across multiple cohorts, providing increased statistical significance and helping to account for random variation within team's cohesion and skill level. I could also foresee a technical and potentially ethical problem with this approach however.

Technically, within Falmouth University Game's Academy, the department where these studies have taken place, teams are not the same each year, they are randomised. This would mean it would be impossible to track the same 'team', although potentially individuals could be tracked and seeing if they implemented CD pipelines of their own accord in future teams could also be a revealing statistic to look at, as it would likely give a measure of *perceived* value, rather than actual value.
The potential ethical problem could be if the initial results

point towards a CD pipeline effecting a students grade, how is that dealt with. Does the experiment carry on as planned, potentially wilfully disadvantaging particular groups of students? While I am uncertain if this would be considered an ethical problem by an ethics board, it could likely provide a moral problem for those involved in running the study. A potential solution could be that those running the experiment would have to be kept in the dark about it's results while it was ongoing to prevent and bias or conflict of interest. While this would certainly require due consideration, it is not a concern for this initial study.

Another limitation is that of resources available. Ideally, every team which a CD pipeline deployed would have support to help customise the pipeline, supporting the team in building custom processes and writing robust unit tests. This would likely give the most 'accurate' imitation of a CD pipeline under industry conditions, however we are unable to provide that level of support due to the research team being a singular individual whom has other commitments, and also lacks the experience to guide teams in writing unit tests specifically.

C. Sampling

Random sampling within Falmouth University Games Academy game's development teams. Aim is to sample as much of the population as possible, including team's across all four years. Mix of random and convenience sampling - any team within the GA is eligible, but will mostly be recruiting teams by speaking to teams in the ga as that's the easiest way to communicate. Will likely also get email sent to all students asking them to participate.

Population is split into four substrata, first years, second years, third years, masters students. Participant teams will then be evenly split between experimental groups who will be given the CD pipeline and control groups. which team's are control groups will be random. This random gathering of participants and assigning of control groups is purposefully to help account for random variation in steam skill and cohesion across all potential participant teams.

D. Data management plan

Data shall be collected using Microsoft Forms as it is GDPR compliant out the box. It will then be stored using Microsoft One Drive, another application that is GDPR compliant out the box. Data will be anonymised but matched after it has been collected, with participants names matching to unique ID numbers. The data will then be stored with this ID, rather than anything identifying like a name or email.

E. Data Analysis

All six hypothesis will be analysed using ANOVA with repeated measurements & between factors. This is an expansion of the dependant t test [18] which allows us to utilise the repeat measurements we will be taking. As we will be running this study with two distinct groupings of teams; teams that have access to the CD pipeline, and teams that do not (and thus are in the control group), this allows us to use the 'between factors', rather than 'within factors' test [18].
Having determined an appropriate test type, we can then make use of GPower Software [19], [20], alongside a comprehensive GPower guide [21], to determine an effective sample size. First, we must provide the software with a number of parameters, which at this stage will be estimates based on our expected results, although after our data has been gathered we will revisit this and calculate these values in order to calculate the actual power of our results. The variables, along with the values we will be providing, are as follows.
Effect Size f: This is the expected effect size we expect to see from our intervention. We are using a value of 0.4 here as this is the recommended value for a large effect size [22], and as we are investigating practical relevance we care far more about large effects than effects that would be too small to provide practical relevance.

α error probability: This will remain at the default value of 5% as this is our probability of getting a type I error, also known as a false positive [23].

Power: As this study is intended as an exploratory study, we will be accepting a power of 0.8. This will enable us to say with an 80% certainty that we will not get a Type II error, also known as a false negative [23].

Number of groups: We will have two groups, our control group who have no access to a CD pipeline, and our experimental group who are provided with the CD pipeline.

Number of measurements: We are planning on collecting eight sets of measurements, one every two weeks from January through to March.

Correlation among repeated measures: We will be using the default value of 0.5 here. This is the recommended value for a moderate to high correlation [21] which is to be expected. We will not be using a higher value as we expect some level of variation due to the project moving through a significant part of its lifecycle during the period we are monitoring, as well as variation due to team's becoming more accustomed to working together and moving through the four stages of team forming [24]. This will be especially prevalent in first and second year teams, as they are more likely to of not worked together due to how Falmouth University Game's Academy operates.

Using these values, we receive a required sample size of thirty participants (see figure 1). While this is realistic for hypothesis regarding individuals, see hypothesis 3 and 4, this will be more challenging for the rest of the hypothesises as they relate to teams as a whole, meaning we would require 30 *teams*, not individuals. This is potentially possible, but will require the participation of two thirds of the entire population of Falmouth University Game's Academy.

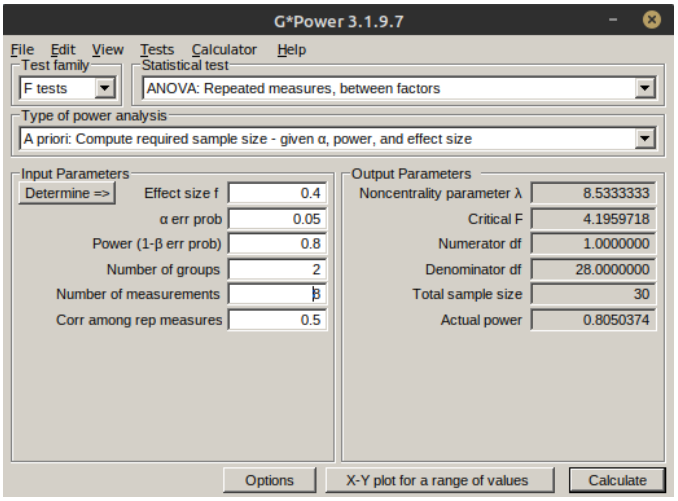

Fig. 1. Screen capture from GPower software showing sample size calculation for an *ANOVA: repeated measurements, between factors* test

F. Ethical Considerations

Due to the nature of this research, there are minimal ethical considerations that need to be taken into account. The participants will not be exposed to any potential risks outside of what they would experience under normal circumstances, although an effort has been made to keep the feedback form fairly condensed. This is to minimise any increased stress that the commitment of having to fill out the form may cause.

V. Appendix

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

VI. Artifact

briefly describe artifact make a brief note on quality assurance link to git repo show off any screenshots or code excerpts

More interested in what and how at this stage

two repo link - one with artifact to easily show off then give link to show it embedded in main project

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
Continuous delivery or continuous deployment? Scope as deployment would need to include itch integration w/ butler, although it looks relatively simple to set up. Auto-upload 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

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] K. B. M. B. A. van Bennekum Alistair Cockburn Ward Cunningham Martin Fowler James Grenning Jim Highsmith Andrew Hunt Ron Jeffries Jon Kern Brian Marick Robert C. Martin Steve Mellor Ken Schwaber Jeff Sutherland Dave Thomas, "Manifesto for agile software development."

[6] W. W. Royce, "Managing the development of large software systems: concepts and techniques," in *Proceedings of the 9th international conference on Software Engineering*, pp. 328–338, 1987.

[7] D. Cohen, M. Lindvall, and P. Costa, "An introduction to agile methods.," *Adv. Comput.*, vol. 62, no. 03, pp. 1–66, 2004.

[8] C. Keith, *Agile Game Development with Scrum*. Upper Saddle River, NJ : Addison-Wesley, 2013.

[9] M. Futter, *The gamedev business handbook: how to build the business you'll build games with*. London: Bithell Games, 2017.

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

[11] M. V. Mäntylä, B. Adams, F. Khomh, E. Engström, and K. Petersen, "On rapid releases and software testing: a case study and a semi-systematic literature review," *Empirical Software Engineering*, vol. 20, no. 5, pp. 1384–1425, 2015.

[12] S. H. Kan, *Metrics and models in software quality engineering*. Addison-Wesley Professional, 2003.

[13] 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.

[14] 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.

[15] 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.

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

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

[18] Carvadia, "What is the difference between a within-subjects anova and a between subjects anova?."

[19] F. Faul, E. Erdfelder, A.-G. Lang, and A. Buchner, "G* power 3: A flexible statistical power analysis program for the social, behavioral, and biomedical sciences," *Behavior research methods*, vol. 39, no. 2, pp. 175–191, 2007.

[20] F. Faul, E. Erdfelder, A. Buchner, and A.-G. Lang, "Statistical power analyses using g* power 3.1: Tests for correlation and regression analyses," *Behavior research methods*, vol. 41, no. 4, pp. 1149–1160, 2009.

[21] D. S. Collingridge, "How to use gpower."

[22] J. Cohen, "A power primer.," *Psychological bulletin*, vol. 112, no. 1, p. 155, 1992.

[23] S. McLeod, "What are type i and type ii errors?."

[24] B. W. Tuckman, "Developmental sequence in small groups.," *Psychological bulletin*, vol. 63, no. 6, p. 384, 1965.