An Investigation into the Effectiveness and Impact of a Continuous Delivery Pipeline upon University-Level Game Development Teams

Student ID: 1907993

***Abstract*—Many small game development teams, especially student teams, face significant challenges in their awareness and understanding of scope, in conducting regular playtesting, and in building rapport and trust with publishers and stakeholders. Continuous Delivery (CD) is a technique designed to help with this by automating the creation of builds. This increases reliability through increased testing, the transparency and accountability of a team, as well as encouraging regular analysis of the pace of development and the project scope. While this is the theory, there exists minimal empirical research to verify this, so this paper will provide an initial, data-driven study into the effects of a CD pipeline upon a student game development team. This has been done by gathering two sets of teams, introducing one set to a CD pipeline, and having both sets fill out a questionnaire every two weeks. This has allowed a comparison to be made between teams using, and teams not using, a CD pipeline.**

I. Introduction

The intention of this paper is to act as an initial investigation into the effects of a Continuous Delivery (CD) pipeline on student development team's. Student team's often have a problem with project scope and reliability of delivering regular builds, which the reported benefits of CD should help with. As such, this study will consist of a case study of a number of student team's within Falmouth University Games Academy, with half the participating team's being provided with a CD pipeline and data being gathered throughout the study by survey. Through this study we will seek to provide some initial answers to whether a CD pipeline can positively effect developer & project supervisor confidence, improve developers understanding of the project's state, and the appropriateness of project scope.

II. An analysis of Continuous Delivery

A. What is Continuous Delivery?

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. This practice is core to the agile methodology, in fact being the first point on the agile manifesto [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 that 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.

III. Case Studies & 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?, and
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, the literature review mentioned above 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, p.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, p.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].

One of the few pieces of literature based upon industry experience is an analysis of the introduction of a CD pipeline into Paddy Power's software development process by Lianping Chen [10], a senior engineer at the company. As acknowledged by the author, even since the paper's release in 2015 this is still one of the only available papers based upon practical experience, rather than purely academic theory. This insight revealed that there were 6 main benefits, but also a number of challenges not addressed in other papers. The benefits were as to be expected from other academic papers; A reduction in cycle time, improved rate's of user feedback, and subsequent increase in developing features that provide the most value to the user. There was also an increase in build reliability, before the introduction of the CD pipeline "Priority 1 incidents caused by manual-configuration mistakes weren't uncommon." [10, p.51], and product quality improved significantly, as the increased level of regular testing reduced the number of open bugs for the application "by more than 90%" [10, p.53]. These are clearly valuable in an industry setting, but could also be valuable to a student development team for whom time is very limited, with a full development time of only 9 months.
The challenges discussed, however, have not been previously mentioned in the academic literature. These challenges were; organisational - the team developing the CD pipeline needed access to resources across the company, requiring negotiation with multiple teams. This need forced a culture shift within the company, with leadership actively working to reduce barriers between team's and improve collaboration. While this shift can be expected to have further benefits, is is noted that this was the largest challenge faced. There were also challenges with existing QA processes - with the cycle time of several months a QA process that delays a release for several days is tolerable, but when there is a cycle time of only days this becomes unacceptable.
These challenges would suggest that, while CD has benefits for team's of all sizes, newer or smaller team's are likely to have an easier time adopting them. Newer team's will have fewer existing processes that need to be reworked, while smaller team's will lack a lot of the organisational problems of perceived area's of control. This could potentially make student team's, which are relatively small and usually newly formed, ideal candidates for the adoption of CD, potentially even ideal as test team's for new or experimental pipelines.

Given this lack of papers relating to CD pipelines within the game's industry or game's education, further papers within this review will be based around CD implementation in the academic setting of software development. This is a parallel field, but one which notably consists much more heavily of programmers writing 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 and interact with a CD pipeline, another case where there is a clear need for further study and literature. An interesting avenue of exploration could be the expansion of the concept of unit tests to fields other than programming. Tests to ensure models are within polygon count boundaries and materials are set up correctly, or perhaps even simulated players that play through levels to check for bugs [13].

One paper, a retrospective of a semester-long course designed

to introduce student's to DevOps and CI/ CD published in 2020 [14], goes into a significant level of detail on how their course was structured and assignments were run. There is, however, an unfortunate lack of reflection on the practical impact of this course, on whether any practices taught were adopted by students during other modules or projects. There is some positive feedback mentioned from students, reportedly in a survey "students commented that they found the assignments beneficial" [14, p.88], although we are provided with no insight into the data set as a whole or even how common this sentiment was amongst the feedback, so this statement is not reliable. This paper gives minimal other insight into the student experience for this course, so is of minimal use for our purpose.

Potentially the most valuable paper in our research has been by Stephan Krusche & Lukas Alperowitz [15], a data based analysis of the introduction of CD practices in a software development course based around multi-customer projects. The course runs over only three months, but works with industry representatives so student's are working on projects with real clients to real deadlines, rather than artificial 'mock' briefs. It once again emphasises the benefit provided by CD of receiving continual feedback from user's, in their words "It enables the idea of continuous user involvement... early in the development process" [15, p.338]. Interestingly, from their analysis of the course they show not only use of CD but the use of other agile principles as well, such as a preproduction stage to build knowledge [15, p.338], macro design [16, timestamp 28:20], and requirement analysis, the presence of which adds the the credibility of the course being designed with agile & industry practice in mind. There are also multiple allusions to the implementation of communities of practice [8], a system to disseminate knowledge on specific topics between cross-discipline development team's, on page 388 & 389.
Moving to the paper's conclusion, There are two main statistics which seem to stand out. First, is that the number of build's ran overall increased by +429% [15, p.342], while the percentage of builds that succeeded rose from 74% to 94%. The second statistic that seems most of note for our study is the indication by student's that they would use version control, CI, and/ or CD in future projects. With over 70% of participants stating they would at least likely apply these concepts to future projects [15, p.342], and even only looking at CI/ CD this was still around 70%, this would lend weight to their earlier statistic that only 20% of student's found no use from any of the taught practices. Even ignoring version control within this calculation, still a similar percentage perceived no value. This gives some good, initial data off of which to base further studies, and could be confirmed with more certainty given a larger sample size.

IV. Research Question & Hypothesis

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. From this proposal, we can draw the research question.

Will the introduction of a continuous delivery pipeline to university-level game development team's have a positive impact on the team's confidence, and on the supervisor's confidence in the team, improving the team's awareness and understanding of the current state and scope of the project?

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, compared to team's not using a CD pipeline. (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

V. Artifact

The artifact to be produced will be the Continuous Delivery pipeline that will be supplied to the experimental team's. This will be designed to take advantage of the version control software used within Falmouth University Games Academy, Github Enterprise [17]. This pipeline will be built using Github Actions [18], an automation system built into Github, and GameCI [19], a CI/ CD api designed for automated build and testing of Unity projects. The pipeline shall consist of 5 main stages. These are detailed below, and the pipeline flow is shown in the Appendix, figure 2. The source code is included in the Appendix, listing 2 and listing 3, as well as being available along with this dissertation at https://github.com/FDonovan98/Dissertation. The pipeline was developed in a standalone project at https://github.falmouth.ac.uk/Glass-Nomad-Games/Testing-Workflows, and can be seen being used in a live project at https://github.falmouth.ac.uk/Glass-Nomad-Games/Silverback, however these links will only be viewable by faculty of Falmouth University Games Academy due to them being hosted on an internal server. The five stages of the pipeline are listed below:

Generate daily pull request: At midnight each night the main branch is compared against the stable branch. If main is ahead of stable then a pull request is created.

Unit tests: Any tests included with the project are then run. These could include frame rate benchmarking, weapon testing, or spawn testing.

Generate Build: Once all unit tests pass, a build is then created using game-ci [19].

Upload Build: After creation, the build is uploaded to the repository as a Github artifact [20]. This could also be expanded to include automatic uploading to Steam, as done on the live project above, using the Steamworks SDK [21] or to Itch using Butler [22], but is potentially out of scope for the current project.

Merge pull request: If the build is created and uploaded successfully, the opened pull request is then merged and closed.

A. Development and Quality Assurance

The development of this artifact followed an Agile [5] and iterative approach, breaking the problem into multiple stages and solving them one at a time, rather than trying to tackle the entire pipeline immediately. Due to the nature of Github Actions, it was not possible to write unit tests as they could only be run through Github itself. This lead to the use of trial and error testing, making small incremental changes and checking viability, before continuing to iterate.

This iterative approach, as well as unit tests, were also used when developing the scripts that parsed questionnaire results from Microsoft Teams, and the script that then performed data analysis on this parsed data. Evidence of unit testing can be seen in parsecsv.py, listing 4, with the function `Verify()` ↪ . This function parses specific test files and then compares the generated file against the expected result, which is a file which was created and parsed manually. If these files match, then a confirmation is printed to the command line, otherwise an error is printed to the command line.

VI. Research Methodology

A. Experimental Design

Participant's for this study will be current students from the Falmouth University Games Academy who are involved in ongoing, academic-year long multidisciplinary game development projects. The selection of this population is both convenience and as a measure for controlling environmental variation, as all student team's within the study will be taken from the same population. There may, however, be relevant substrata within this population that are not being accounted for: the academic year the development team is in. This is discussed more in the first paragraph of the Limitation's section.

Participating team's will then be split into two groups; experimental - those who will be provided with a CD pipeline, and control, whom will not be provided with a CD pipeline. Each team member will then be sent a survey every two weeks to complete, with at least one question relating to each hypothesis.
Each team also has an academic supervisor, a member of the faculty who meets with the group once a week to discuss how their project is going and view their latest progress. It is worth noting that the academic supervisor is the one who marks the team for their game development module, and while they can provide advice and guidance they have no agency over any decisions the team makes, except in exceptional circumstances. This member of staff will also be given a survey every two weeks which is different to the student one, in order to explore hypothesis 1 and 2.

The study will be run over 11 weeks from January 10th 2021 to March 28th 2021, at the conclusion of which the data will be analysed using the *ANOVA: repeated measurements, between factors* method. See the section titled Data Analysis, & figure 1, for an explanation and discussion on sample size.

B. Limitations

The primary limitation in this study is the population size available to draw from, due to needing 2/3 of the population to take part in order to achieve statistical significance (see the section on Data Analysis for further details). This could be solved either by increasing the duration of the study, or increasing the population size available to draw from. If this study were to be expanded and run again, I would ideally like to increase the study duration and follow team's from their first year all the way through to graduation. This would allow half of the teams to be introduced to the CD pipeline immediately, and then their results, feedback, and attitudes across all three years could be tracked against the control groups. This could also be done at the same time across multiple cohorts, providing increased statistical significance and helping to account for random variation between different cohorts general level of skill and experience.
The second option would be to increasing the population available to the study by reaching out to and running the study with multiple institutions, however I would caution against this without due consideration. Due to the potentially significant environmental variation between institutions in staffing, course format, team agency, and an incredibly high number of other things, team's from different institutions would fall within different strata, meaning the data gathered could not reliably be treated as a single data set.
If the study were to be run over an increased time span as suggested above, I can foresee a technical and potentially ethical concern that would require due consideration when designing the study. These are discussed in Section X, as they are not a concern for this initial study.

Another concern with the available population is the existence of four substrata within this population. The four main substrata present are the four academic years within Falmouth University Games Academy, three undergraduate years and a Masters year. The environmental variation within these substrata will be minimised due to all students being of the same institution, however, there will still be variation across several factors. This variation will primarily be in experience,

which will, broadly, increase with the academic year of the student team.

A second limitation is that of resources available to those conducting the study. Ideally, every team with a CD pipeline deployed would have support to help customise the pipeline, supporting the team in building custom processes and writing robust unit tests to be executed within the pipeline. This would likely give the most 'accurate' imitation of a CD pipeline under industry conditions, however it is not possible to provide that level of support to all experimental team's. This is 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 effectively.

A team's stage within the four stages of team development [23] also has the potentially to vary significantly between the substrata discussed previously. First year teams are quasi-randomised, with the intention of giving team's an even distribution of skills while putting students into a group with new people. This means the chance of a first year team having never worked together before is very high. Entering their second year student's are given more agency to choose who they work with in a development team, and then for third year student's are given complete agency to create team's with whomever they wish. This mean's that in second and especially third year team's are more likely to of worked with at least some members of the team before, allowing them to progress much quicker through Tuckman's model, or skip stages entirely to reach the most productive later stage's quicker. This could mean the variation between substrata is larger than it otherwise would be.

Given these limitations, this study is intended as an initial investigation and starting point for further research, rather than aiming to provide definitive answers.

C. Sampling

A mix of convenience & voluntary response sampling of Falmouth University Games Academy team's will be used, as all team's are eligible to take part. Convenience sampling will be used by speaking to team's that are within the main Games Academy building, a studio space where a majority of team's work. In theory, this has the potential for giving access to the entire population, although given the current circumstances with Covid [24] some team's may have opted to work fully remotely, so would not be excluded by this sampling method. As such, accompanying this method will be a batch of voluntary response sampling. An email shall be sent to all current student's within the department, briefly detailing the study and asking students to take part.

Participant teams will then be randomly assigned as experimental groups, who will have access to the CD pipeline, until half of all participant teams are in the experimental group. All other team's will then be assigned as control groups, who will not have access to the CD pipeline. This random assigning of teams to the experimental group is purposefully done in order to help account for random variation in a team's skill and cohesion.

D. Data management plan

Data shall be collected using Microsoft Forms as it is GDPR compliant out the box. Data will then be anonymised, with participants names being replaced with unique ID numbers in order to protect participants and prevent any bias during data analysis, before being stored using Microsoft One Drive, another product that is GDPR complaint out the box.

E. Data Analysis

All six hypothesis will be analysed using the *ANOVA: repeated measurements, between factors* method. This is an expansion of the dependant t test [25] 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, experimental and control, this allows us to use the 'between factors', rather than 'within factors' test [25].
Having determined an appropriate test type, we can then make use of GPower Software [26], [27], alongside a comprehensive GPower guide [28], 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 statistical power of our results. The variables, along with the values we will be providing, are as follows.

Effect Size f: This is the 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 [29], and as we are investigating practical relevance we care far more about large effects than small effects.

α *error probability:* This will remain at the default value of 5% as we want this low as this is our probability of getting a type I error, also known as a false positive [30].

Power: As this study is intended as an exploratory study, we will be accepting a power of 80%. 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 [30].

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 six sets of measurements, one every two weeks from January through to March.

Once data has been collected, it can be parsed using a python script and analysed in R. The data collected in this way will consist of ratings, from 1 to 10, of the team's confidence in the

accuracy of their scope, how well they understand the current state of the project, and how often the scope for the project is refined. The supervisor's confidence both in the team being able to present a working build at their next meeting and in being able to finish everything within their scope, rated from 1 to 10, will also be analysed with this method.
There will also be the option for participants to expand on these ratings and give more verbose feedback, which could later be analysed using sentiment analysis [31].
Below is an excerpt of sample code in R that can be used to conduct a one way ANOVA test on results loaded in from a csv file. This has been produced using the cited guides as reference [32], [33].

```
# Read's in data from a .csv file
# Declares whether columns contain
    ↪ quantitative or categorical data
mockdata <- read.csv("mock_data.csv",
    ↪ header = TRUE, colClasses = c("
    ↪ factor", "factor", "numeric"))

#Performs a one way ANOVA analysis
# Uses confidence as dependant variable
# and experimental as the
# independent variable
oneway <- aov(confidence ~ experimental
    ↪ , data = mockdata)

# Summarises the results,
# including F & P value
summary(oneway)
```

Listing 1. Sample R code for performing a one way ANOVA on data imported from a csv file

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 [28] 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 development [23]. 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 Games Academy operates.

Using these values, we receive a required sample size of 32 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 32 *teams*, not individuals. This is potentially possible, but will require the participation of two thirds of the entire population of Falmouth University Games Academy.

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

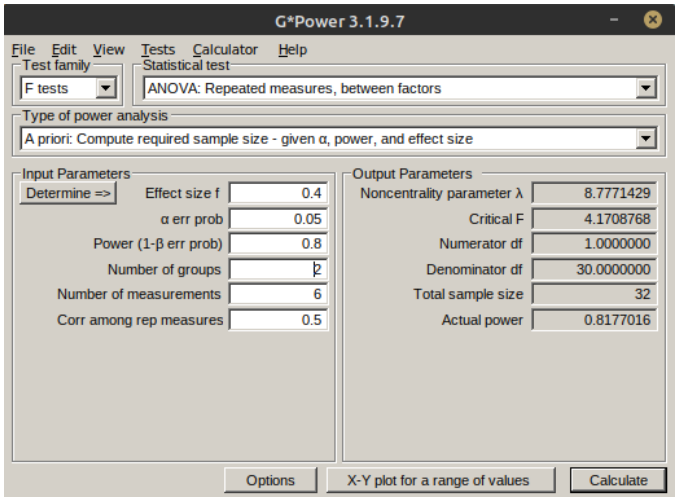

Fig. 1. Screen capture from GPower software showing sample size calculation for an *ANOVA: repeated measurements, between factors* test

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. Personal data will be collected, which will be stored and handled in compliance with GDPR regulations as discussed in the Data Management section. Care has also been taken to ensure no participant rights have been violated with respect to the Declaration of Helsinki [34] or Nuremburg Code [35]
The artifact itself is of minimal risk, as it is collecting no personal information and does not have the potential to be weaponised.
Given these two factors, the project has been deemed medium risk and has such has been signed off by the research project Supervisor and Head of Subject, Dr Michael Scott, in accordance with Falmouth University's ethics policy [36].

VII. Data Analysis

Data for this research was analysed using radian version 0.5.12 and R version 3.6.3.

Data for each hypothesis was analysed using repeated measures ANOVA [25] using the R function **aov** [37]. Three different ANOVA tests were then carried out, a one-way test against if the group was using the pipeline, a two-way test with the specific Team as a modifier, and another two way test with the team as a modifier and with the individual person having an interaction effect. An Akaike Information Criterion (AIC) [38] test was then used to evaluate which of these models best fits our data.

The most appropriate model identified above, which for all hypothesise was the one-way ANOVA, was then used with the R function **effectsize::omega_squared** [39] to calculate the effect size. Cohen's f was originally going to be used to compare effect sizes, however after consultation with more experienced academics, η^2 was chosen instead. After further research, this was replaced with ω^2, as in the

R documentation and elsewhere [39]–[41] it is recommended to use this over η^2 as it is reportedly less biased, especially for small sample sizes [41]. As our sample size is only 25 results, this seems appropriate. The boundaries that will be used for contextualising the reported effect size are below,, which is based upon the recommended heuristics from Andy Field [42].

Very Small -	$0 >$	ω^2	$<= 0.01$
Small -	$0.01 >$	ω^2	$<= 0.06$
Medium -	$0.06 >$	ω^2	$<= 0.14$
Large -	$0.14 >$	ω^2	

The ω^2 value for many of these hypothesis was negative, and while this would normally be rounded to 0.00, this has been left as-is due to recommendations in the cited paper [43].

Several of our initial hypothesis, 1 and 2, related to the supervisors opinions and confidence with regards to their team. This data was unable to be collected due to low uptake by teams & even lower uptake by the supervisors for those teams. As only seven teams agreed to take part, and of these only two supervisors responded to the questionnaire, it was decided not to pursue these hypothesis and leave them as avenues for future research.

The Results For Each Hypothesis

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, compared to team's not using a CD pipeline: No data collected, as discussed above.

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: While supervisor data was not collected so a comparison can not be made, data regarding the team's confidence in their own scope was recorded. Pr(>F) of 0.94, so there is a 94% chance that the observed effect could be from random chance, which is not statistically significant. F = 0.006, ω^2 = -0.04.

3) A CD pipeline will help developers understand the current state of the project: The one-way ANOVA test resulted in a Pr(>F) of 0.68, so there is a 68% chance that the observed effect could be from random chance, which is not statistically significant. F = 0.175, ω^2 = -0.04.

4) A CD pipeline will help developers to always know why the work they are doing is being done: Pr(>F) of 0.707, so there is a 71% chance that the observed effect could be from random chance, which is not statistically significant. F = 0.145, ω^2 = -0.04.

5) A team using a CD pipeline will refine their scope more often: Pr(>F) of 0.0284 *, so there is a 3% chance that the observed effect could be from random chance, which is statistically significant. F = 5.5, ω^2 = 0.16, which is a large effect size.

6) A team using a CD pipeline will do more playtesting: Pr(>F) of 1, so there is a 100% chance that the observed change could be from random chance, which is not statistically significant. F = 0, ω^2 = -0.04.

VIII. Discussion of results

With an initial uptake of 7 teams and 34 people, and 27 responses to the questionnaires over four lots of data collection, the limited usefulness of the gathered results is no surprise. It is encouraging, however, to see some statistically significant results. Given the variation in the other results it is possible that further investigation with a larger sample size could still yield interesting results, as we are unable to rule out it does have an effect with our sample size. Assuming these results are representative, see Section IX for a discussion on this, we can see that it is very likely there is a relation between whether a team uses a CD pipeline and how often they refine their scope, and that this relationship has a significant effect size.

The lack of results for the other hypothesis may be an indication that there are other compounding factors that effect the results. For example, for hypothesis 6 it would be expected that a team using a CD pipeline would do more playtesting as they always have a latest build available, however it may be that the produced builds are not actually being played, despite being generated daily.

IX. Issues and Limitations

A. A Limited Sample Size

The initial analysis indicated a required sample size of 32, which with 34 sign ups was achieved. However over the four times the questionnaire was sent out to participants only 17 unique responses were received, with 25 responses received in total. This small sample size is one of the reasons the results in this study may not be representative of the general population.

This lack of response is believed to have stemmed from a lack of physical interaction between the researcher and the participants. This lack of rapport and accountability, combined with the alternative to in person communication being sending emails to accounts which are rarely checked, meant that few signed up and fewer still responded after signing up. There was potentially also a lack of incentive for teams to take part due to the limited resources available to the researchers. It was not possible to offer anything such as small financial compensation for participants time, so this is something which, if implemented in future research, may improve response rate. As more research is produced, those results could also be used as incentive if they show a positive correlation between the use of a CD pipeline and team performance.

B. Unbalanced Sample Pool

Only a single team involved in the experiment had the pipeline implemented, rather than half of the teams involved. This was due to the difficulty with communication as discussed above. This means that any variation could have been due to random variation, and only having a single experimental team means this wouldn't of been averaged out across the sample.

X. Conclusion and Future Work

These initial results indicate some promise, and as discussed within Section III should provide an initial empirical, data driven reference for further work. Due to random variation that has been unaccounted for with the small sample size, see Section IX for further details, results here should likely be taken as inspiration for further research, rather than used as evidence of definite effect.

For further research, it would be recommended to work with teams over a longer period, most likely a full academic year, and focus on building rapport and familiarity initially to improve the amount of data collected, as discussed in Section IX. It would also be recommended to work closely with each teams leader and/ or supervisor, to act as a single point of accountability for each team, and as someone who can also remind the team to fill in the questionnaire regularly. If research is again conducted in an academic setting, it would likely be beneficial to work closely with staff in the team recruitment stage to identify teams that would be most appropriate, likely those that are already proven to be reliable. This could, however, produce bias where the pipeline is only tested with teams who fit certain criteria so are not representative of the population as a whole, so this approach would need careful consideration. As mentioned briefly in Section VI-B, there are also two potential concerns that would require due consideration, one technical, and one ethical.

For the technical concern, game development team's within Falmouth University Games Academy are not the same each year, they are quasi-randomised. This would mean it would be practically impossible to track the same 'team' across multiple years. Potentially, individuals who have taken part in the study could be tracked, even if their team was not selected as an experimental group in the following year. Seeing if these individuals then implement their own CD pipelines could be a revealing statistic to look at as it could give a measure of *perceived* value, if not actual value. If this did occur regularly however then control groups would effectively be becoming experimental groups, potentially making a statistically sound comparison harder.

The potential ethical concern is due to this initial study pointing towards potential benefits and advantages for teams that utilise CD. This could mean that teams not involved in the study, or selected as control groups, could be disadvantaged both in terms of the product they produce and the grade they receive. While one solution could be to make the potential benefits clear to all students when recruiting, so they are able to give informed consent, a better solution may be a redesign of the experimental methodology. If the research is run over a longer period, such as an entire academic year as recommended above, then data could be gathered for the first half of the year with no teams having the CD pipeline. Then, all teams are introduced to the CD pipeline at the same time, and further data is collected. This research only did not follow that methodology due to time constraints, and it would likely be better at accounting for random variation and would present less ethical problems as all teams are treated equally.

XI. Reflective Addendum

The results of the research were overall disappointing, suffering from a severe lack of engagement, resources, and time. As discussed in Section X, this study would benefit from a longer time frame to collect results, recruit participants, and provide support with the implementation of the CD pipeline.

Testing and quality assurance for the artifact was light due to the nature of it - github actions can only be tested on a repo through Github, so unit tests are not possible. This lead to a lot of trial and error testing, and the requirement of having to upload the code to github, then wait for the tests to run on the github actions server, was frustratingly slow at times. For future development, it is possible that a small but relatively high spec server set up specifically for testing Github Actions code would help to speed up this process significantly. This would allow the tests to run much faster, as well as prevent tests being stuck in queue behind longer, in-production Github Action processes.

The other written code, the data analysis in R and the python parsing, was heavy on iterative testing but light on structured unit testing. This was due to both of these scripts being exploratory in nature, and were they to be rewritten a comprehensive test suite could likely be made. With the simplicity of both of these scripts that is likely to be of limited use however, and writing the tests would likely take longer than writing the actual code. While if they were actually of significant importance, for example a key piece of software in a production environment, that would be worth the time investment, for their current use it would likely be wasted effort, especially given the tight time constraint the dissertation as a whole is produced under.

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. [Online]. Available: https://www.atlassian.com/continuous-delivery/principles/continuous-integration-vs-delivery-vs-deployment

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

[4] Atlasian. (2021) Bamboo best practice - using stages. [Online]. Available: https://confluence.atlassian.com/bamboo/bamboo-best-practice-using-stages-388401113.html

[5] K. Beck, M. Beedle, A. Van Bennekum, A. Cockburn, W. Cunningham, M. Fowler, J. Grenning, J. Highsmith, A. Hunt, R. Jeffries *et al.* Manifesto for agile software development. [Online]. Available: https://agilemanifesto.org/principles.html

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

[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] C. Gordillo, J. Bergdahl, K. Tollmar, and L. Gisslén, "Improving playtesting coverage via curiosity driven reinforcement learning agents," *arXiv preprint arXiv:2103.13798*, 2021.

[14] M. Hills, "Introducing devops techniques in a software construction class," in *2020 IEEE 32nd Conference on Software Engineering Education and Training (CSEE T)*, 2020, pp. 1–5.

[15] 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*, ser. ICSE Companion 2014. New York, NY, USA: Association for Computing Machinery, 2014, p. 335–343. [Online]. Available: https://doi.org/10.1145/2591062.2591163

[16] M. Caerny. Gb d.i.c.e. summit 2002 - mark cerny. [Online]. Available: https://www.youtube.com/watch?v=QOAW9ioWAvE

[17] Github. Github for enterprises. [Online]. Available: https://github.com/enterprise

[18] Github. Github actions. [Online]. Available: https://github.com/features/actions

[19] GameCI. Gameci. [Online]. Available: https://game.ci/

[20] Github. Storing workflow data as artifacts. [Online]. Available: https://docs.github.com/en/actions/advanced-guides/storing-workflow-data-as-artifacts

[21] Steam. Uploading to steam. [Online]. Available: https://partner.steamgames.com/doc/sdk/uploading

[22] Itch.io. Butler. [Online]. Available: https://itch.io/docs/butler/

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

[24] J. Gallagher. New covid variant: How worried should we be? [Online]. Available: https://www.bbc.co.uk/news/health-59418127

[25] Carvadia. What is the difference between a within-subjects anova and a between subjects anova? [Online]. Available: https://carvadia.com/what-is-the-difference-between-a-within-subjects

[26] 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.

[27] 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.

[28] D. S. Collingridge. How to use gpower. [Online]. Available: http://www.mormonsandscience.com/gpower-guide.html

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

[30] S. McLeod, "What are type i and type ii errors?" [Online]. Available: https://www.simplypsychology.org/type_I_and_type_II_errors.html

[31] N. Altrabsheh, M. Cocea, and S. Fallahkhair, "Sentiment analysis: towards a tool for analysing real-time students feedback," in *2014 IEEE 26th international conference on tools with artificial intelligence*. IEEE, 2014, pp. 419–423.

[32] A. Kassambara. Repeated measures anova in r. [Online]. Available: https://www.datanovia.com/en/lessons/repeated-measures-anova-in-r/

[33] R. Bevans. Anova in r: A step-by-step guide. [Online]. Available: https://www.scribbr.com/statistics/anova-in-r/

[34] M. D. Goodyear, K. Krleza-Jeric, and T. Lemmens, "The declaration of helsinki," 2007.

[35] Nuremberg Code, "The nuremberg code," *Trials of war criminals before the Nuremberg military tribunals under control council law*, vol. 10, no. 1949, pp. 181–2, 1949.

[36] Falmouth University. Research & innovation integrity & ethics. [Online]. Available: https://www.falmouth.ac.uk/research/support/research-ethics-integrity/#observed-policies

[37] datacamp. aov: Fit an analysis of variance model. [Online]. Available: https://www.rdocumentation.org/packages/stats/versions/3.6.2/topics/aov

[38] R. Bevans. Akaike information criterion — when & how to use it. [Online]. Available: https://www.scribbr.com/statistics/akaike-information-criterion/

[39] datacamp. cohens_f: Anova effect size (omega squared, eta squared, epsilon squared). [Online]. Available: https://www.rdocumentation.org/packages/parameters/versions/0.2.0/topics/cohens_f

[40] S. Glenn. Omega squared: Definition, spss. [Online]. Available: https://www.statisticshowto.com/omega-squared/

[41] C. Albers and D. Lakens, "When power analyses based on pilot data are biased: Inaccurate effect size estimators and follow-up bias," *Journal of Experimental Social Psychology*, vol. 74, pp. 187–195, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S002210311630230X

[42] A. Field, *Discovering statistics using IBM SPSS statistics*. sage, 2013.

[43] K. Okada, "Negative estimate of variance-accounted-for effect size: How often it is obtained, and what happens if it is treated as zero," *Behavior Research Methods*, vol. 49, no. 3, pp. 979–987, 2017.

XII. Appendix

```
name: Open main to stable pull request
on:
  schedule:
  # Runs at 1am Tuesday through Saturday
  - cron: "0 1 * * 2-6"

  workflow_dispatch:

jobs:
  merge:
    name: Create pull request
    runs-on: azure-docker
    steps:
      # Checkout (without LFS)
      - name: Checkout repository
        uses: actions/checkout@v2
      # Git LFS
      - name: Create LFS file list
        run: git lfs ls-files -l | cut -d' ' -f1 | sort > .lfs-assets-id
```

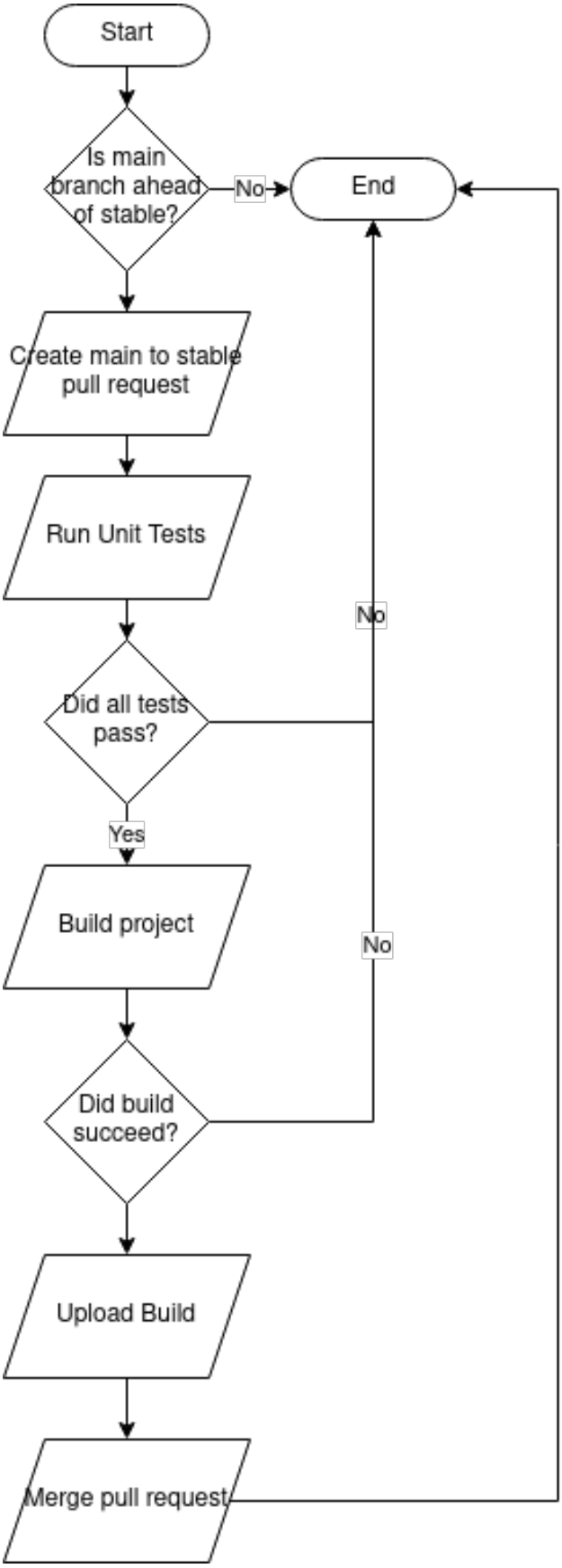

Fig. 2. A diagram showing the flow of the Continuous Delivery system

```
      - name: Git LFS Pull
        run: |
          git lfs pull
          git add .
          git reset --hard

      - name: Create pull request
        id: create-pr
        uses: HDonovan96/pull-request@v2.6.2
        with:
          source_branch: "main"
          destination_branch: "stable/daily"
          github_token: ${{ secrets.REPO_
              ↪ SCOPE_TOKEN }}
```

Listing 2. Github Actions code to open a pull request from 'main' branch to 'stable/daily' branch at midnight each weeknight

```
# This is a basic workflow to help you get
    ↪  started with Actions

name: Build, test, & merge pull requests
    ↪ to stable

# Controls when the action will run.
on:
  # Triggers the workflow on push or pull
      ↪ request events but only for the
      ↪ main branch
  # push:
  # branches: [ main ]
  pull_request:
    branches: [ stable/* , main]
    types: [ opened, reopened, synchronize]

  # Allows you to run this workflow
      ↪ manually from the Actions tab
  workflow_dispatch:
  workflow_call:

# A workflow run is made up of one or more
    ↪  jobs that can run sequentially or
    ↪ in parallel
jobs:
  # This workflow contains a single job
      ↪ called "build"
  build:
    name: Build for ${{ matrix.
        ↪ targetPlatform }}
    runs-on: azure-docker
    strategy:
      fail-fast: false
      matrix:
        targetPlatform:
          - StandaloneWindows64 # Build a
              ↪ Windows 64-bit standalone.
    steps:
      # Checkout (without LFS)
      - name: Checkout repository
```

```
  uses: actions/checkout@v2
# Git LFS
- name: Create LFS file list
  run: git lfs ls-files -l | cut -d' ' -f1 | sort > .lfs-assets-id
- name: Git LFS Pull
  run: |
    git lfs pull
    git add .
    git reset --hard

# game-ci couldn't find _temp/ so manually create it
- name: Create temp
  run: |
    echo $USER
    mkdir _temp/

# Test
- name: Run tests
  uses: game-ci/unity-test-runner@v2
  env:
    UNITY_EMAIL: ${{ secrets.UNITY_EMAIL }}
    UNITY_PASSWORD: ${{ secrets.UNITY_PASSWORD }}
    UNITY_SERIAL: ${{ secrets.UNITY_SERIAL }}
  with:
    githubToken: ${{ secrets.GITHUB_TOKEN }}

- name: Git reset
  run: |
    git reset --hard
    git clean -fdx

- name: Build Project
  uses: game-ci/unity-builder@v2
  env:
    # Unity details stored using github secrets
    UNITY_EMAIL: ${{ secrets.UNITY_EMAIL }}
    UNITY_PASSWORD: ${{ secrets.UNITY_PASSWORD }}
    UNITY_SERIAL: ${{ secrets.UNITY_SERIAL }}
  with:
    targetPlatform: ${{ matrix.targetPlatform }}
    buildName: silverback

- name: Upload Build To Actions
  uses: actions/upload-artifact@v2
  with:
    name: Build-${{ matrix.targetPlatform }}
    path: build/${{ matrix.targetPlatform }}

- name: Enable automerge
  uses: alexwilson/enable-github-automerge-action@main
  with:
    github-token: "${{ secrets.GITHUB_TOKEN }}"
```

Listing 3. Github Actions code that runs on every opened Pull Request. Runs unit tests builds the project and then uploads it to steam

```
from ast import Not
# from asyncio.windows_events import NULL
import csv
import hashlib
from random import random
import filecmp

# Reads data from .csv file, including only wanted columns
def ParseData(filePath, validColumns, columnHeaders):
    # First set of data will need to have column headers
    # Following sets will not
    if(columnHeaders == False):
        parsedText = []
    else:
        parsedText = columnHeaders

    y = 0

    with open(filePath, newline='') as csvfile:
        file = csv.reader(csvfile, dialect='excel')
        for row in file:
            x = 0
            temp = []
            for word in row:
                # y != 0 to ignore .csv headers in favour of custom headers
                if (y != 0):
                    # Ignore any columns we don't want results from
                    if(x in validColumns):
                        temp.append(word)

                x += 1

            # y != 0 to ignore .csv headers in favour of custom headers
            if (y != 0):
                parsedText.append(temp)
            y += 1
```

```
    return parsedText

# Test function to strip all data from a .csv, regardless of wanted columns
def SimpleParse(filePath):
    y = 0
    parsedText = []
    with open(filePath, newline='') as csvfile:
        file = csv.reader(csvfile, dialect='excel')
        for row in file:
            temp = []
            for word in row:
                if (y != 0):
                    temp.append(word)
            if(y >= 1):
                parsedText.append(temp)
            y += 1

    return parsedText


def WriteDataToOutputFile(filepath, data):
    with open(filepath, 'w') as outputFile:
        writer = csv.writer(outputFile)
        writer.writerows(data)

    return 0


# Hash email to obfuscate it but keep entries by the same person related
def HashColumn(data, columnIndex):
    y = 0
    for row in data:
        if (y != 0):
            row[columnIndex] = hashlib.sha1(
                row[columnIndex].encode('utf-8')).hexdigest()
        y += 1

    return data

# Marks whether data has come from an experimental or control group
def MarkDataAsExperimental(isExperimental, data, header):
    if (header == False):
        start = 0
    else:
        data[0].insert(0, header)
        start = 1
    for i in range(start, len(data)):
        data[i].insert(0, isExperimental)

    return data


def CombineData(dataA, dataB):
    return dataA + dataB

# Uses generated lookup table to assign each participant to the correct team
def SetTeams(data, header):
    data[0].insert(1, header)
    for i in range(1, len(data)):
        hasSetValue = False
        for j in range(0, len(teamLookupTable)):
            for k in range(1, len(teamLookupTable[j])):
                if (data[i][1] == teamLookupTable[j][k]):
                    hasSetValue = True
                    data[i].insert(1, teamLookupTable[j][0])

                    if(hasSetValue): break

            if (hasSetValue): break

            if (j == len(teamLookupTable)-1):
                print("ERROR: " + data[i][1] + " not present in teamLookupTable. Assigning random value for " + header)
                data[i].insert(1, str(random()))

    return data

# Reads data from experimental .csv file & control .csv file
# Then marks them as experimental or control
# Then outputs the data to another file
def CreateParsedSupervisorData(controlDataPath, experimentalDataPath, outputDataPath):
    validColumns = [3, 10, 16]
    columnHeaders = [['id', 'buildConfidence', 'scopeConfidence']]

    experimentalSupervisorData = ParseData(experimentalDataPath, validColumns, columnHeaders)

    experimentalSupervisorData = MarkDataAsExperimental(True, experimentalSupervisorData, 'isExperimental')
```

```
    validColumns = [3, 10, 16]
    columnHeaders = False

    controlSupervisorData = ParseData(
        controlDataPath, validColumns,
            ↪ columnHeaders)

    controlSupervisorData =
        ↪ MarkDataAsExperimental(
        False, controlSupervisorData, False)

    combined = CombineData(
        ↪ experimentalSupervisorData,
        ↪ controlSupervisorData)
    combined = SetTeams(combined, 'team')
    combined = HashColumn(combined, 1)
    combined = HashColumn(combined, 2)

    WriteDataToOutputFile(outputDataPath,
        ↪ combined)

def GenerateLookupTable(filePath):
    parsedData = SimpleParse(filePath)
    return parsedData

# Maps a verbose value in a specific
    ↪ column to a numeric value.
# This is needed for the data analysis in
    ↪ R.
def ConvertVerboseToNumeric(data,
    ↪ columnIndex, verboseArray):
    y = 0
    entryConverted = True
    for row in data:
        if (y != 0):
            entryConverted = False
            for i in range(0, len(
                ↪ verboseArray)):
                if (row[columnIndex] ==
                    ↪ verboseArray[i]):
                    row[columnIndex] = i
                    entryConverted = True
                    i = len(verboseArray)
        if (not entryConverted):
            print("Error: Verbose entry " +
                ↪ row[columnIndex] + " had no
                ↪ equivalent in verboseArray
                ↪ ")
        y += 1

    return data

# Parses student data, only data relevant
    ↪ to the R data analysis is included
    ↪ in the output file.
def CreateParsedStudentData(
    ↪ controlDataPath,
    ↪ experimentalDataPath, outputDataPath
    ↪ ):
    # Defines columns to be included in the
        ↪ output file
    validColumns = [3, 10, 16, 22, 25, 28]
    columnHeaders = [['id', '
        ↪ scopeConfidence', '
        ↪ stateUnderstanding', '
        ↪ contributionUnderstanding', '
        ↪ scopeFrequency', '
        ↪ playtestFrequency']]

    experimentalData = ParseData(
        ↪ experimentalDataPath,
        ↪ validColumns, columnHeaders)

    experimentalData =
        ↪ MarkDataAsExperimental(
    True, experimentalData, 'isExperimental
        ↪ ')

    # Defines columns to be included in the
        ↪ output file
    validColumns = [3, 10, 16, 22, 25, 28]
    columnHeaders = False

    controlData = ParseData(
        controlDataPath, validColumns,
            ↪ columnHeaders)

    controlData = MarkDataAsExperimental(
        False, controlData, False)

    combined = CombineData(experimentalData
        ↪ , controlData)
    combined = ConvertVerboseToNumeric(
        ↪ combined, 5, ['Once a semester\n'
        ↪ , 'Every other sprint\n', 'Once
        ↪ per sprint\n', 'Multiple times
        ↪ per sprint\n'])
    combined = ConvertVerboseToNumeric(
        combined, 6, ['Only at events (Demo
            ↪ day etc.)\n', 'Every other
            ↪ sprint\n', 'Once per sprint\n'
            ↪ , 'Multiple times per sprint\n
            ↪ '])
    combined = SetTeams(combined, 'team')
    combined = HashColumn(combined, 1)
    combined = HashColumn(combined, 2)

    WriteDataToOutputFile(outputDataPath,
        ↪ combined)

def ParseStudentData():
    # Generates a lookup table, mapping
        ↪ individual participants to their
        ↪ team
    global teamLookupTable
```

```
    teamLookupTable = GenerateLookupTable('
        ↪ Participants.csv')

    # Set file path's
    controlDataPath = 'Dissertation␣Survey␣
        ↪ -␣Student␣Version␣N.csv'
    experimentalDataPath = 'Dissertation␣
        ↪ Survey␣-␣Student␣Version␣P.csv'
    outputDataPath = 'parsed_student_
        ↪ results.csv'
    CreateParsedStudentData(
        controlDataPath,
            ↪ experimentalDataPath,
            ↪ outputDataPath)

def ParseTestData():
    # Generates a lookup table, mapping
        ↪ individual participants to their
        ↪ team.
    global teamLookupTable
    teamLookupTable = GenerateLookupTable('
        ↪ testparticipants.csv')

    # Set file path's
    controlDataPath = 'testdata_N.csv'
    experimentalDataPath = 'testdata_P.csv'
    outputDataPath = 'parsed_test_results.
        ↪ csv'
    CreateParsedStudentData(
        controlDataPath,
            ↪ experimentalDataPath,
            ↪ outputDataPath)


# 'expected_parsed_results.csv' created by
    ↪  hand from test data
def CheckCreatedAgainstExpected():
    if (filecmp.cmp('expected_parsed_
        ↪ results.csv', 'parsed_test_
        ↪ results.csv', shallow=False)):
        print("Files␣verified")
    else:
        print("ERROR:␣File␣did␣not␣verify")

def Verify():
    ParseTestData()
    CheckCreatedAgainstExpected()

# ParseStudentData()
Verify()
```

Listing 4. Python code that parses the results .csv generated by Microsoft Forms and exports only the useful data to a new .csv. User details are also encrypted.

```
# Reads in data from parsed .csv
# Data analysis on test data
data <- read.csv("data_analysis/parsed_
    ↪ test_results.csv", header = TRUE,
    ↪ colClasses = c("factor", "factor", "
    ↪ factor", "numeric", "numeric", "
    ↪ numeric", "numeric", "numeric"))

# Data analysis on student data
# data <- read.csv("data_analysis/parsed_
    ↪ student_results.csv", header = TRUE,
    ↪  colClasses = c("factor", "factor",
    ↪ "factor", "numeric", "numeric", "
    ↪ numeric", "numeric", "numeric"))

library(AICcmodavg)

# Goes through each independent variable
    ↪ and runs data analysis
for (i in 4:8) {
    # as.formula is used so that this code
        ↪ can be generalised enough to work
        ↪  in a loop
    formula <- as.formula(paste0(paste(
        ↪ names(data)[i]), "␣~␣
        ↪ isExperimental"))
    oneway <- aov(formula, data = data)

    formula <- as.formula(paste0(paste(
        ↪ names(data)[i]), "␣~␣
        ↪ isExperimental␣+␣team"))
    twoway <- aov(formula, data = data)

    formula <- as.formula(paste0(paste(
        ↪ names(data)[i]), "␣~␣
        ↪ isExperimental␣*␣team␣+␣id"))
    blocking <- aov(formula, data = data)

    model.set <- list(oneway, twoway,
        ↪ blocking)
    model.names <- c("oneway", "twoway", "
        ↪ blocking")

    # Prints results from data analysis
    print(names(data)[i])
    print(summary(oneway))
    # Calculates effect size
    print(effectsize::omega_squared(oneway)
        ↪ )
    # Check which model fits the data the
        ↪ best
    print(aictab(model.set, modnames =
        ↪ model.names))
}
```

Listing 5. R code to perform ANOVA tests upon results after file is parsed with Python. Best fit analysis is then performed upon these tests.