ALDARUNNER

Report

Submitted by Dave Hoevenaars

Fontys Hogeschool Techniek en Logistiek

Reuver, 22-06-2021

# Management Summary

In this project sequence, the possibilities of automating the testing and the grading of student code through GitHub will be analysed and advised upon. The final goal is to have a clear cut advice which will describe how the teachers can automate grading and testing through GitHub.

This will be the first project sequence, therefore there is no previous work to be referred to, as there will most likely be a follow up of the project sequence, I will suggest a few thing how the project sequence could continue.

As this project is part of a sequence, it is important to clearly define the scope, so that its easier to see what was and was not achieved in this project sequence.

Table of contents

[Management Summary 2](#_Toc76949287)

[1. Introduction 4](#_Toc76949288)

[2. Project Justification 5](#_Toc76949289)

[2.1 Project objectives 5](#_Toc76949290)

[2.2 Project requirements 5](#_Toc76949291)

[2.3 Project scope 5](#_Toc76949292)

[3. Project products 6](#_Toc76949293)

[3.1 Customer quality expectations 6](#_Toc76949294)

[3.2 Customer acceptance criteria 6](#_Toc76949295)

[4. Solution & Results 7](#_Toc76949296)

[4.1 Analysis 7](#_Toc76949297)

[5. Conclusion & Recommendations 17](#_Toc76949298)

[Literature List 18](#_Toc76949299)

[Resources 19](#_Toc76949300)

[Questions 21](#_Toc76949301)

# 1. Introduction

My assignment is consisting of researching and realising a prototype of an automated assignment handling system, which will remove the need for labour intensive testing and also reduce the complexity and possibly the vulnerability level of the current system. This document reflects my findings about this in such a way that it can be easily picked up by whoever continues within this project sequence.

# Project Justification

## Project objectives

The goal of this project is to give the teachers advice on how to utilise the GitHub platform in order to conduct automated testing and grading. In order to do that I will be focussing on GitHub Actions and Classrooms. Once the current situation has been analysed and the *to be* has been designed, the prototype is tested in order to give advice on how to best implement the prototype, taking into account scalability, maintainability and security.

## Project requirements

To finish the project, we need some resources. These will be provided by Fontys or by myself. The things we need are:

- Java assignments

- GitHub Actions

- GitHub Classrooms

- Maven knowledge

- Surefire reports

- JaCoCo, Maven build phase

## Project scope

The scope describes what will be done during this project. All involved parties will know what they can expect during this project. We also describe the exclusions. This makes clear what won’t be done in this project.

In scope

- Analysis of the current situation for building and testing assignments

- Design of the situation to be for building and testing assignments and or exams

- Prototype of building process for exams and assignments

- How to Run Surefire report

- How to Run Pitesting report

- How to Run JaCoCo coverage report

- Advice on how to keep your GitHub actions private

- Advice on how automated correcting of assignments and exams could take place at Fontys

Out of scope

- How to connect Canvas with GitHub classrooms

- How to implement secrets to hide the teacher tests from the student

- How to enable autograding

# Project products

In this chapter the quality of the products will be discussed. Spread over the 18 weeks that are set out for the project there will be a couple of different products that are going to be delivered to the customer. This chapter addresses the quality, the expectations of the customer and the acceptance criteria for the different products.

## Customer quality expectations

In this part the customer’s quality expectations are explained. This topic describes the project deliverables that the customer, expects in this project.

In various meetings it was shared that the expectations of the project are as follows:

- Remain relevant to the time constraints of the project;

- Well documented project, with listed resources for technologies that were applied.

- The advice on how to automated testing and building, must include security from student attempted fraud. The problem with the standard GitHub action solutions is that students would be able to see which code is being run in the GitHub actions defined by the teachers, therefore it would pose a security risk.

## Customer acceptance criteria

The project deliverables and acceptance criteria regarding quality and intended use are described in detail in order to serve as an official agreement between the project stakeholders. This part describes the sufficiently added value of the project to the customer in order to consider this project to be successful.

|  |  |  |  |
| --- | --- | --- | --- |
| Deliverable | Acceptance criteria | Product goal | Intended use |
| As is analysis | Fact-based, transparent | Creating clarity of the current system in place | To be used a reference for developing the To Be Design |
| To be Design | Every feature from the requirement document has been met | Providing the arguments of investment justification to the Fontys board | To create a clear overview of the new assignment handling solution |
| Implementation Advise | Step by step approach | Guideline of the assignment handling solution implementation. Described in steps | This plan is going to serve as the guideline of the next stage in the assignment handling solution of the customer |

Table 1 products

# Solution & Results

In this section the progress that has been made will be discussed such as analysis artefacts and working prototypes.

## Analysis

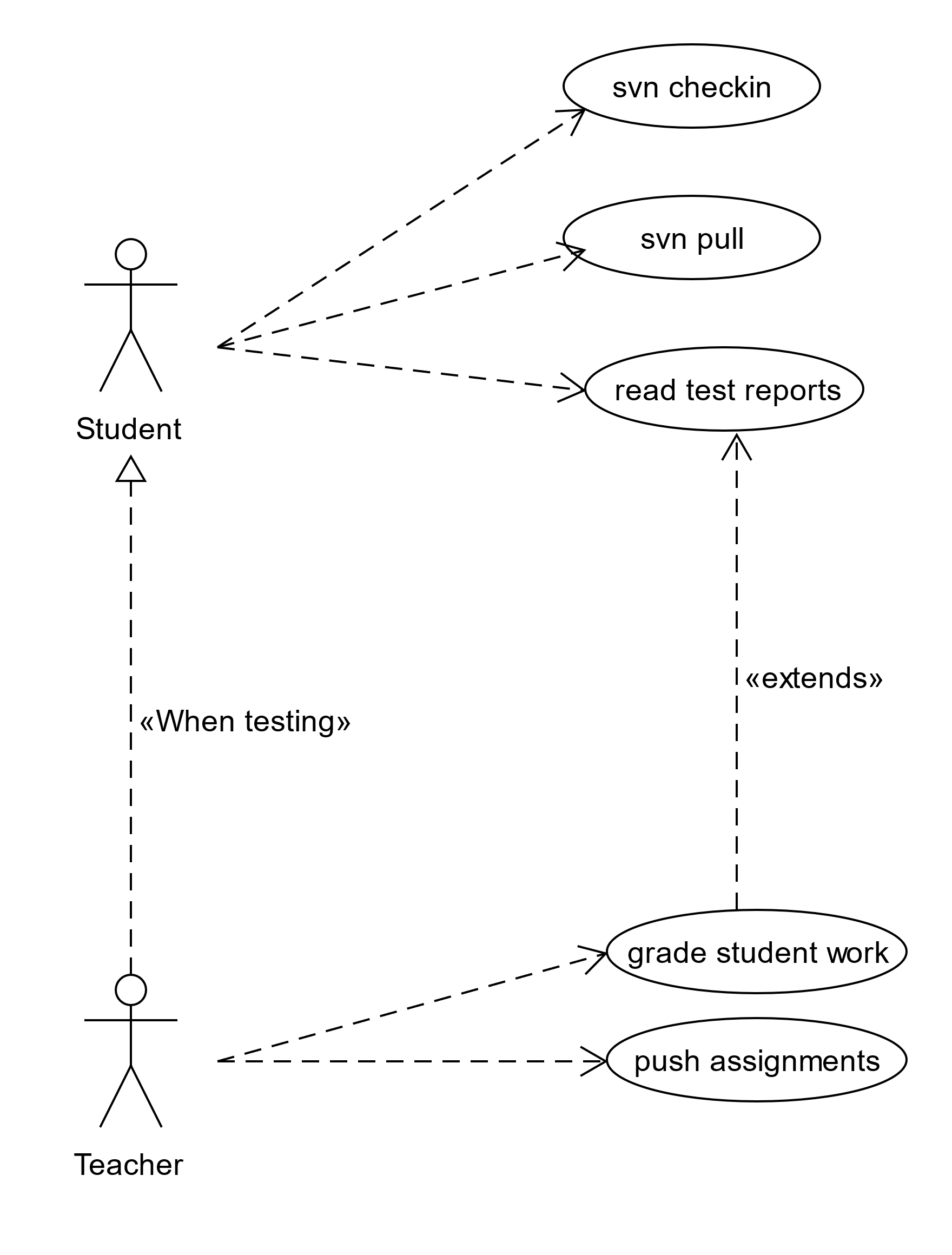


Figure 1 Usecase diagram

Here it is displayed which use cases are available for both user types, please note that the teacher and student overlap in some cases, as is the case with read test reports and grade student work. This is because the grade of the student work is based on the test reports.

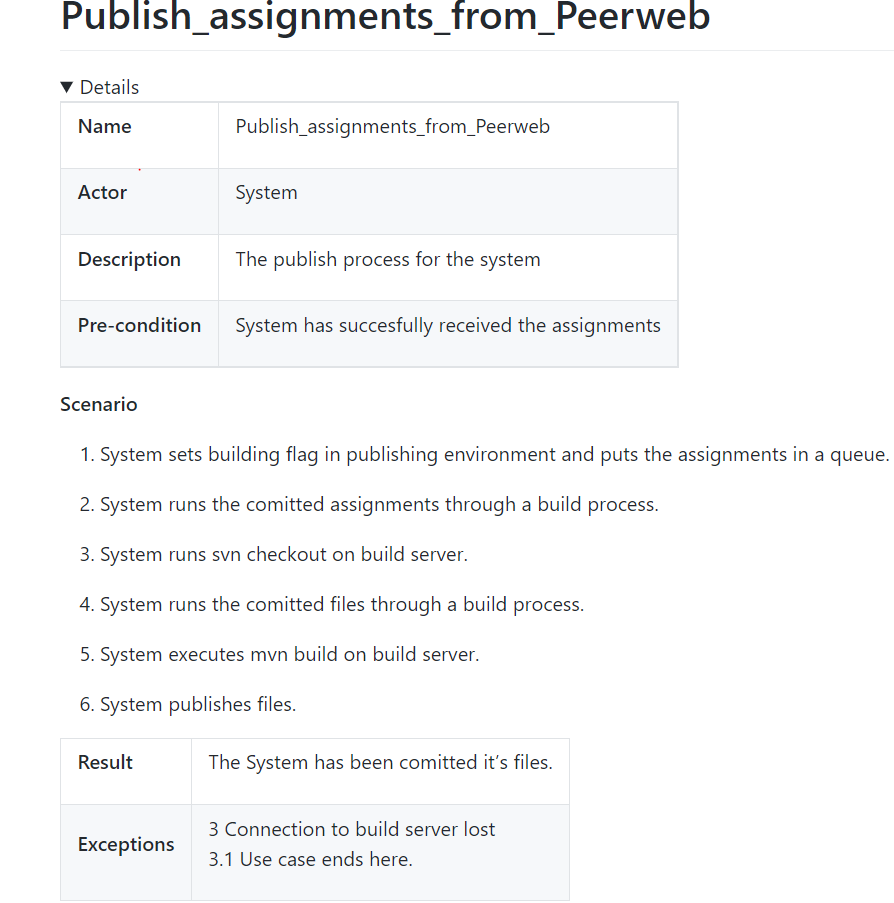


Figure 2 Publish assignment from Peerweb

This usecase scenario represents how the system publishes the assignments from Peerweb

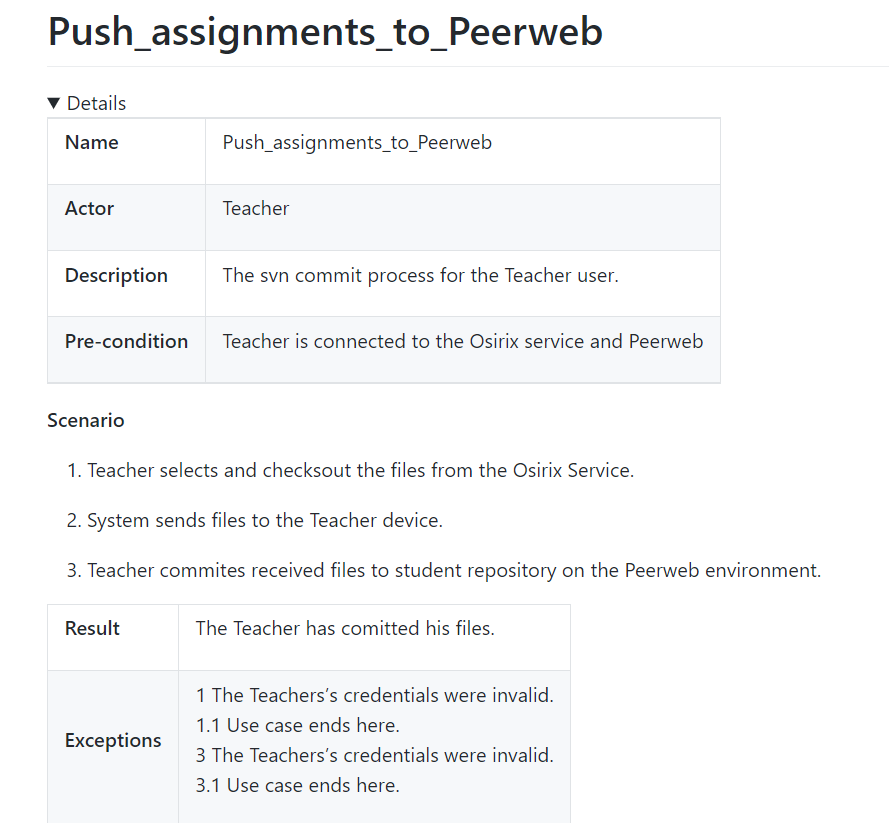


Figure 3 push assignment to Peerweb

This usecase scenario represents how the teacher pushed the assignments to Peerweb

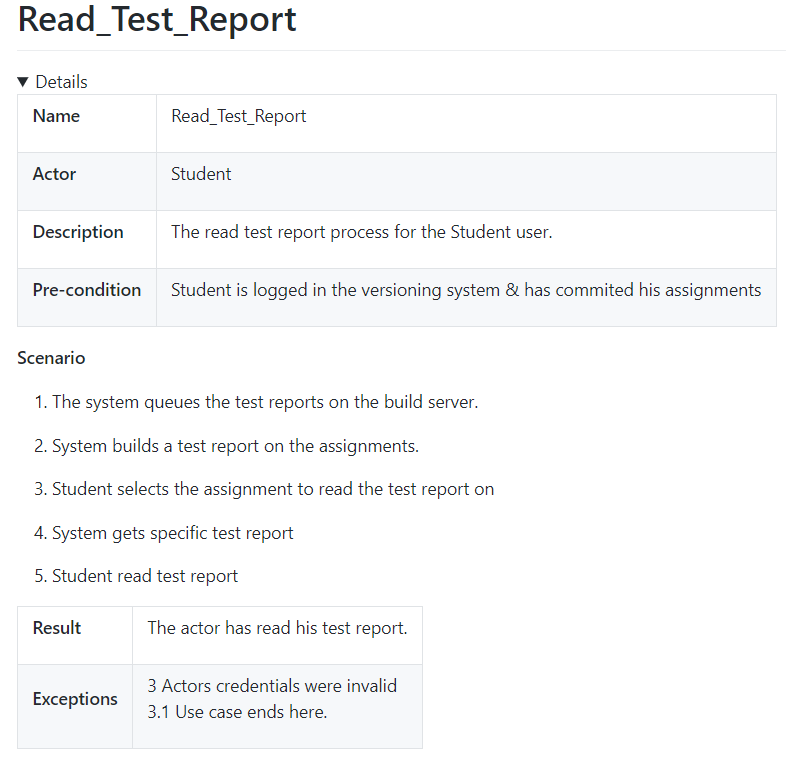


Figure 4 read test report

This usecase scenario represents how the student receives his test report from Peerweb

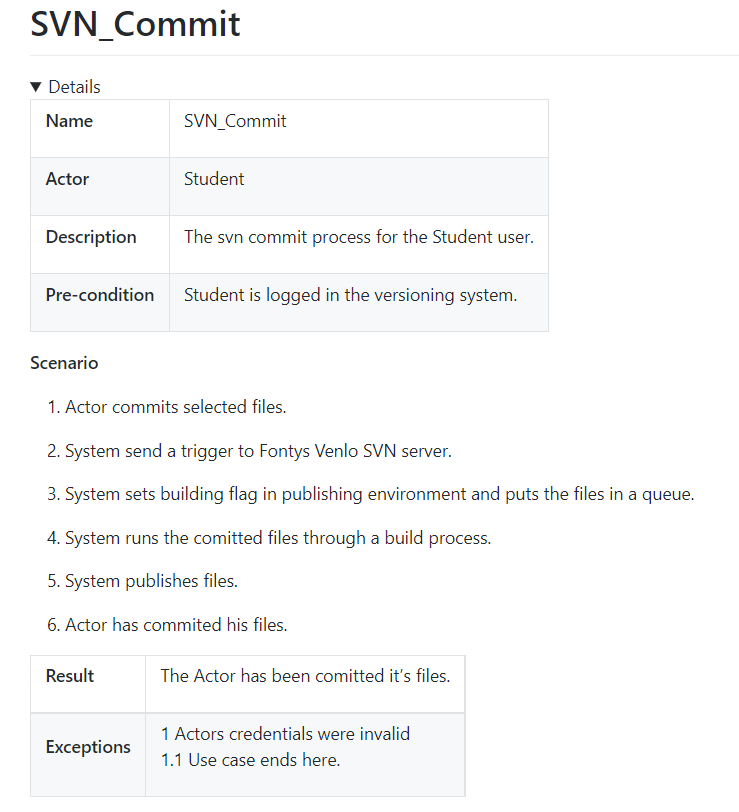


Figure 5 SVN Commit

This usecase scenario represents how the student receives his test report from Peerweb.

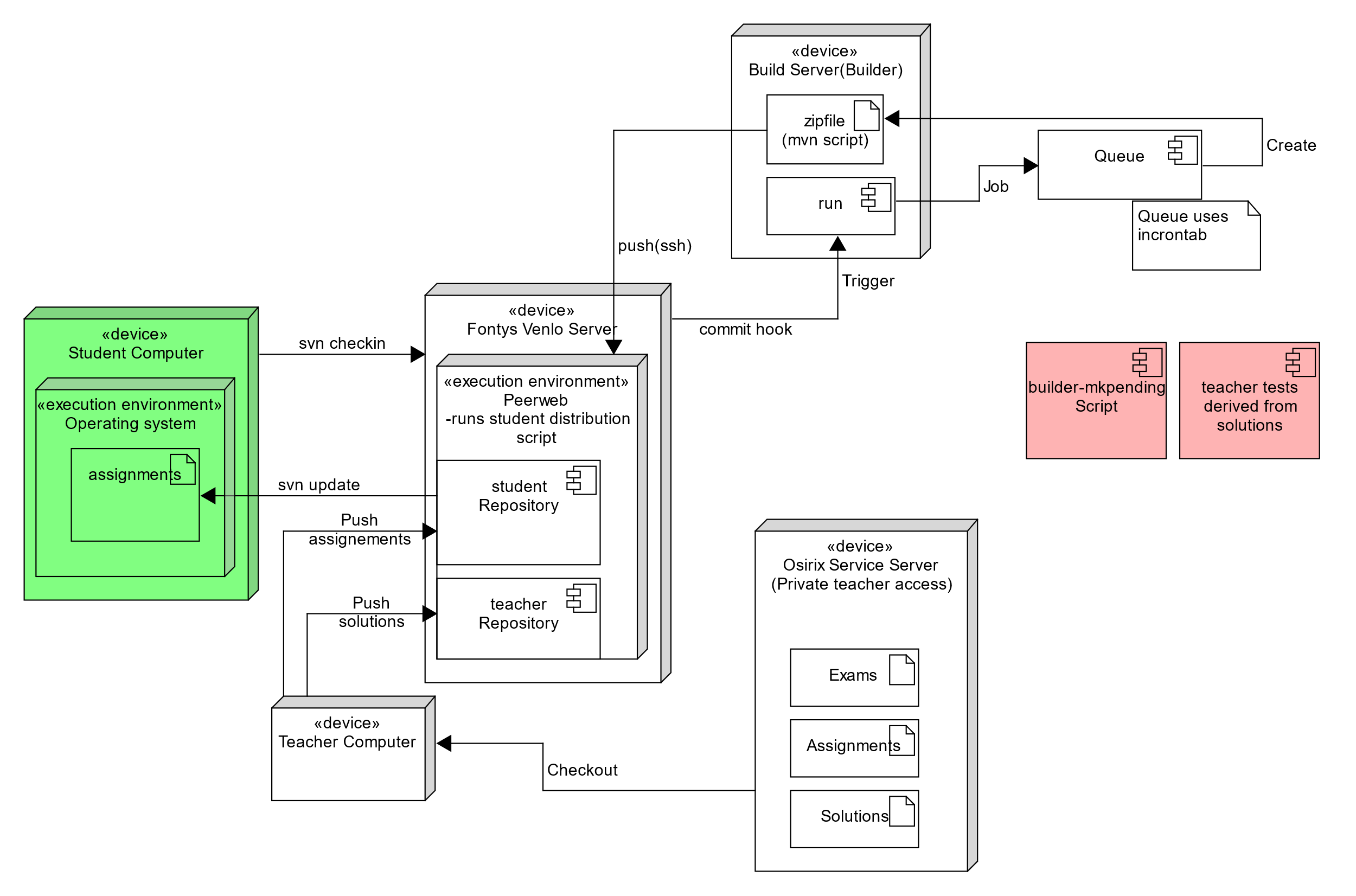


Figure 6 : Deployment Diagram(as is)

Please note that also in this image you can see how the teacher and student use-cases overlap, as the teacher will need to push his assignments to the Subversion repository in order for the student to able to get them. The data flow starts with the assignments being pulled from the Osirix Service Server and being pushed from the teacher, to the student repository. The student is then able to do svn update to get the assignment, do some work and check-in his progress. Then the commit hook will trigger a build job on the build service, which puts the job in to a queue which produces a zip-file.

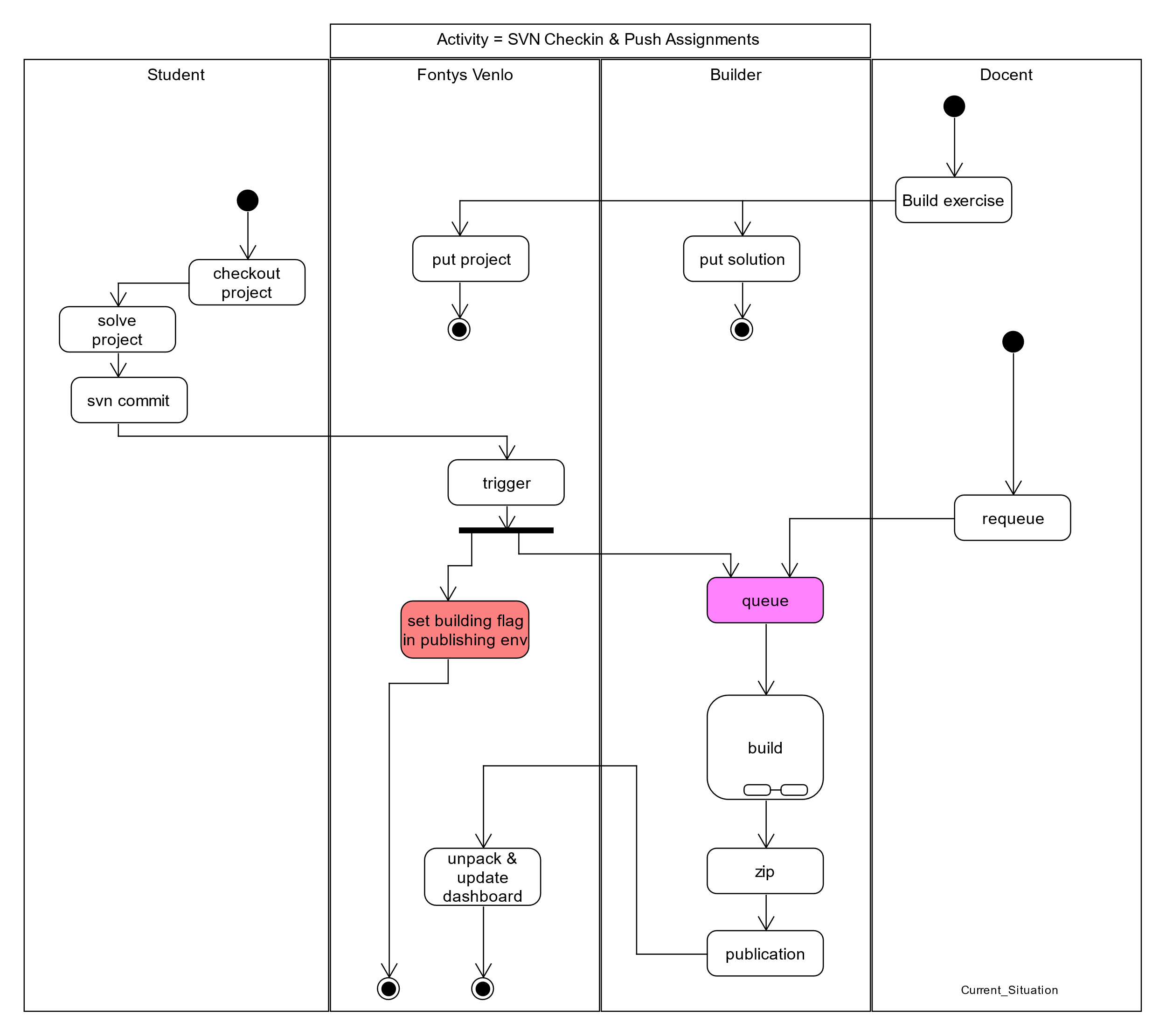


Figure 7 Activity Diagram: AsIs

Here it is shown how the current situation is in terms of handling assignments. The teacher can build an exercise, which will put the project in the Fontys Venlo environment and the solution in the builder. The student can then checkout the project, solve it and commit it, which will trigger a building job and set a building flag in the publishing environment, the building flag takes the shape of an animation on the Peerweb dashboard. The building job is firstly put into a queue and then the building process takes place. The build process itself will be discussed more in dept later. The build process then crates a zip-file which contains the products(reports) of the build process and moves them to the Peerweb publication environment where the student can see how he/she did.

## 4.2 Design

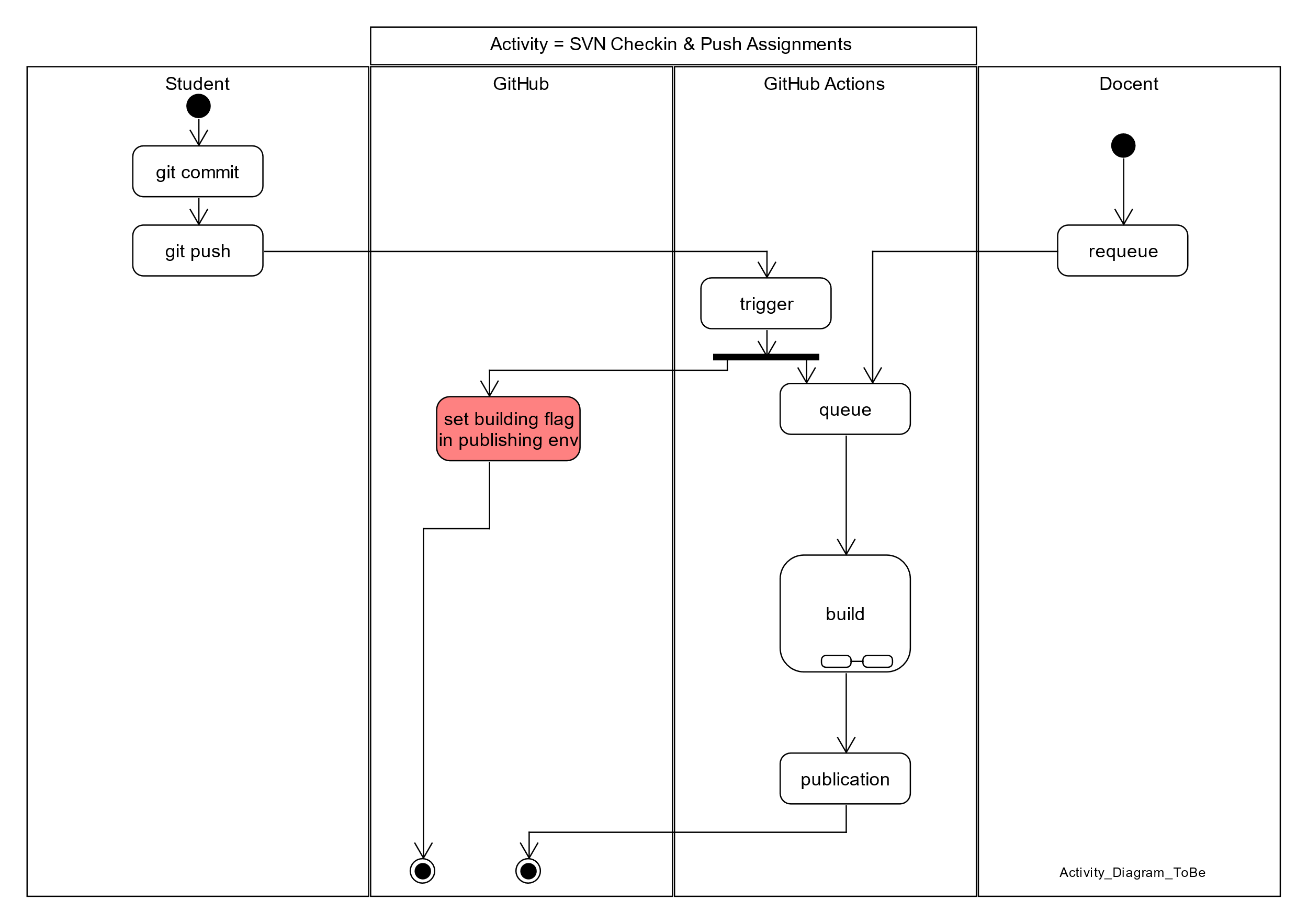


Figure 8 Activity Diagram: ToBe

Here it is shown how the solution is to be, the student will commit & push his work to GitHub, which will trigger a building flag and an action that will be put in the queue. This building flag will signal that the Action is executing. For the build process the same applies to it as it does in the As Is situation. The only difference being that it publishes on GitHub instead of Peerweb

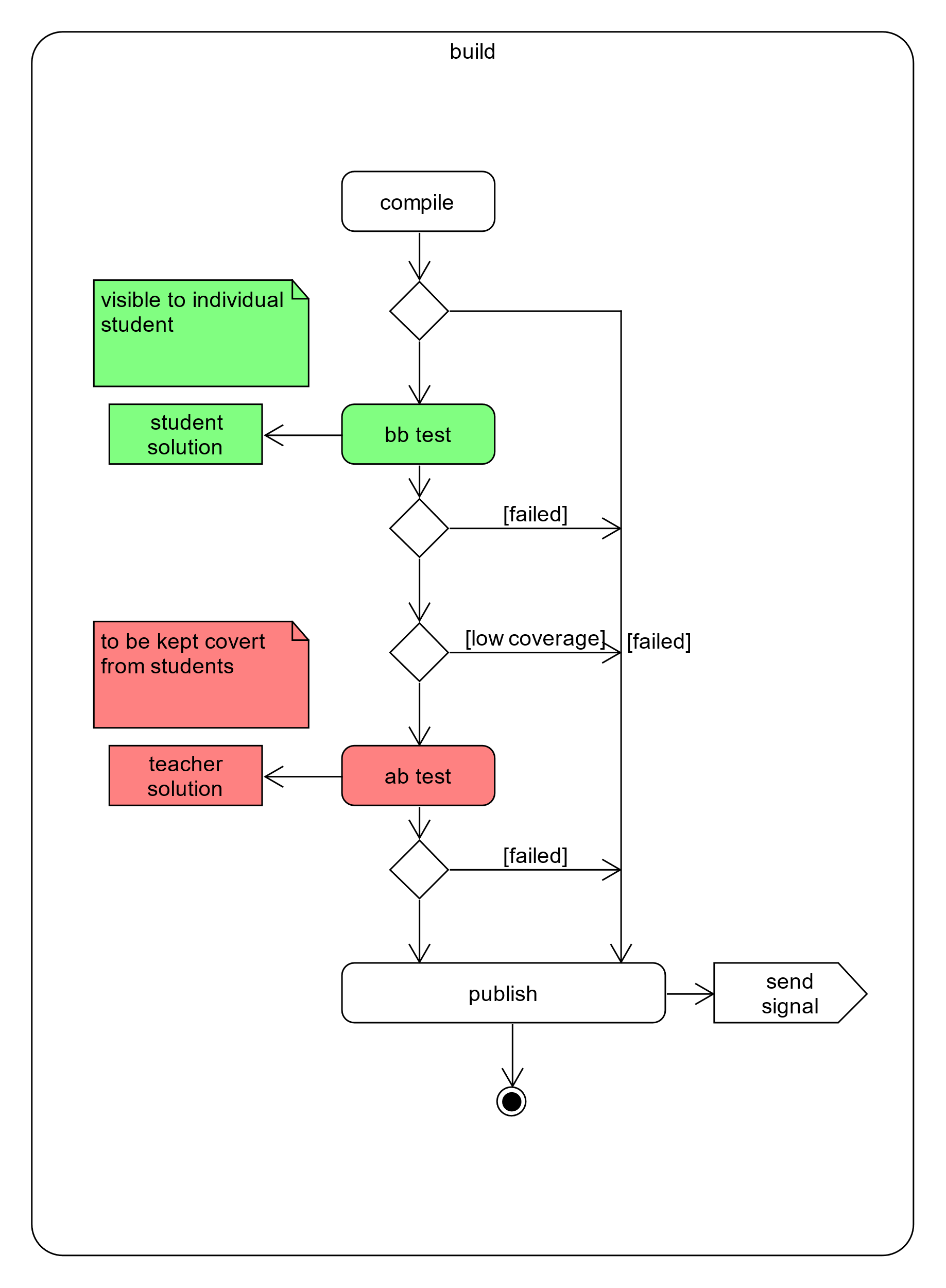


Figure 9 Build details

Here the maven build process is explained, as it is a maven process, it will be possible to mimic it in GitHub Actions. First it will compile the student code, and if successful it will run the student (BB[[1]](#footnote-1)) tests with Surefire. If the BB tests are successful, the maven process will then use JaCoCo to generate a coverage report, if that coverage is below a certain number, the process will go into a fail state, publish the test runs and stop execution. Assuming a successful case, the maven process will then again use Surefire to run the teacher tests on the student code(AB[[2]](#footnote-2)). Once every test has been run it will publish those results and send a signal of some sorts. Some sorts because it can be defined for what kind of signal would be preferred.

Explain AB systems (1st is the testcode, 2nd is the business code)

A means teacher, B means student. So BB is student test on student code etc.

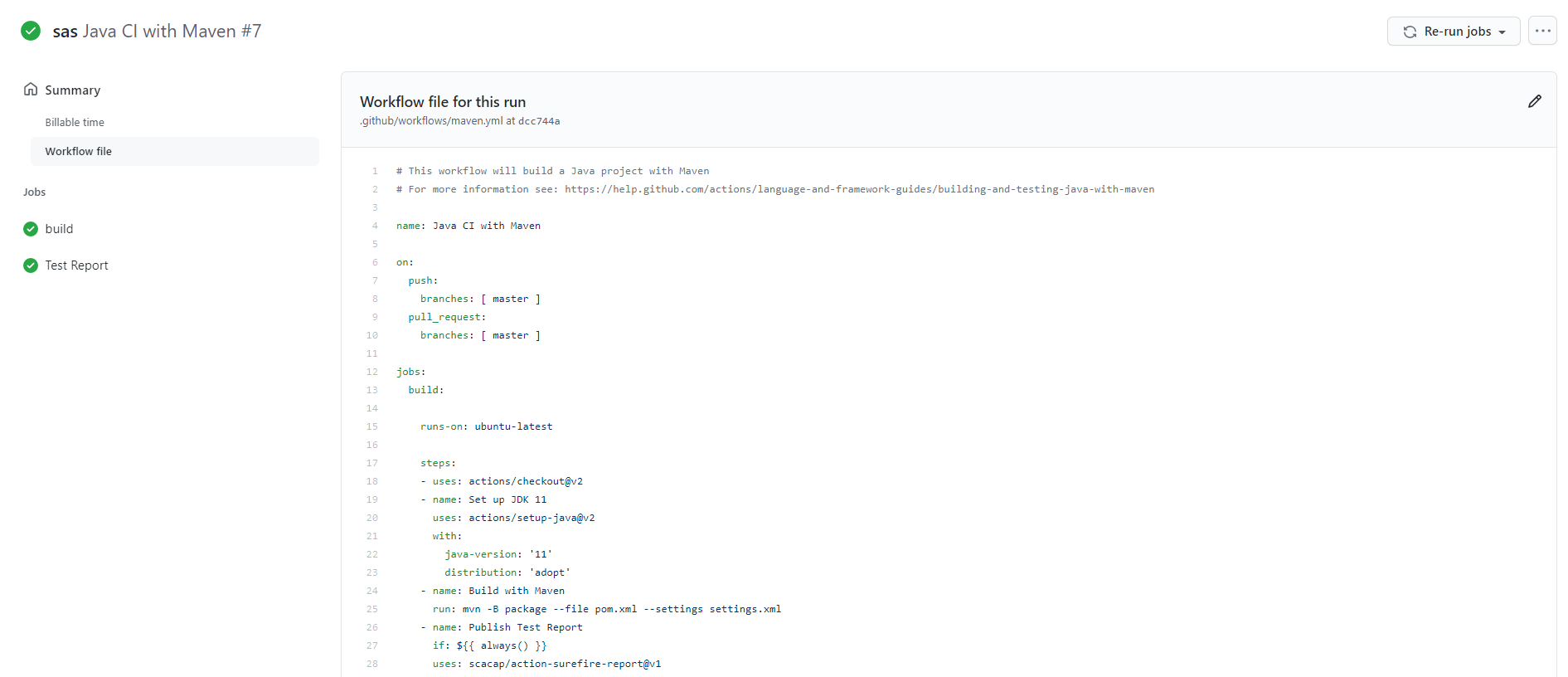


Figure 10 Maven build and Surefire Action

Here it is showed how a Maven build and Surefire Action can be defined, the Maven action is easy to define, you simply click on add work flow, if you don’t have one already, and then you’ll be able to select a Maven action which is already defined and ready to go. As long as you have your maven dependencies set correctly in your repository of course. If that works than you can simply navigate to GitHub marketplace for the Surefire part and the implementation will be listed there. Then it is simply the case of adding it’s dependency in your pom file again, and your good to go.

Also note that in GitHub Maven actions, you can run any maven command that you like, just specify the name of your step and on the line below, which you command you want to run, this is done as follows:

* name: Maven Depenency Tree

run: mvn dependency:tree

By running this maven action, we are fulfilling part of the current automated testing and grading system.

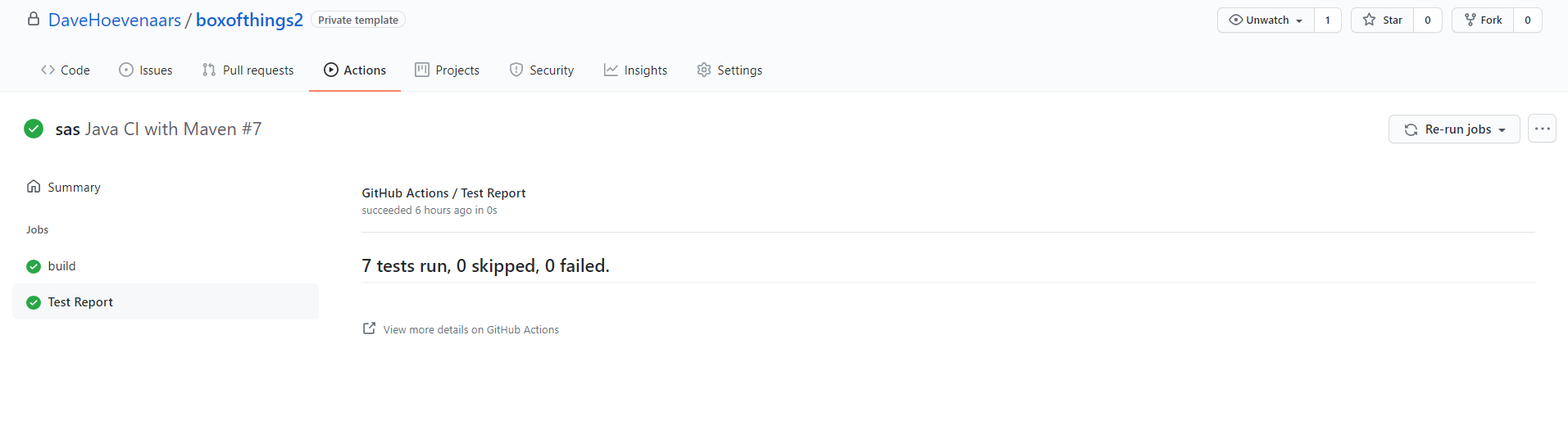


Figure 11 No failed tests Surefire run

This is a run from Surefire without failed tests, if that’s the case you will receive no report at all, just a line saying every test passed.

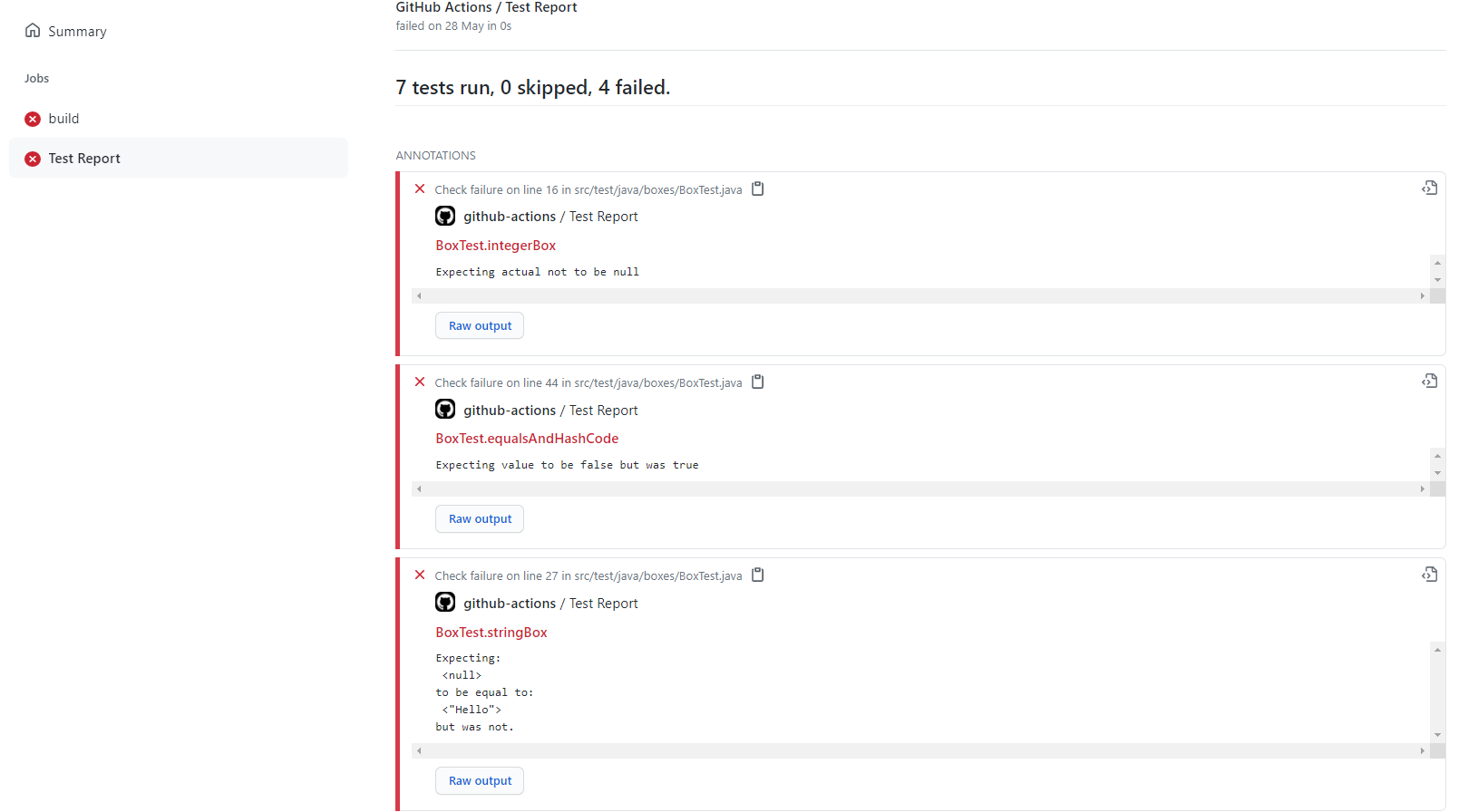


Figure 12 Failed tests Surefire run

When you do fail tests however, you will see a lot of detail, including: what tests failed, why they failed and what the test output was. Also do note that if the tests fail that, Maven does not bother building the solution since the tests did not pass.



Figure 13 JaCoCo Report and Badge Action

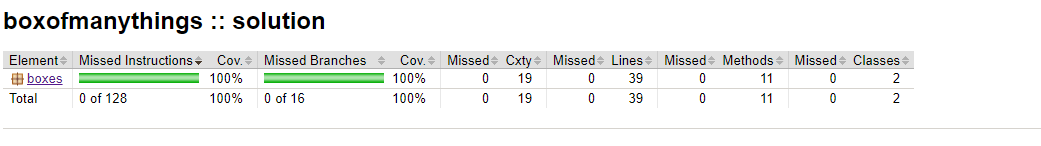
Here we can see how the JaCoCo action is defined, note that you can specify that the target directory for output and even can list an user to forward the result to.

Figure 14 JaCoCo Report run result

This JaCoCo coverage report show the percentage of code covered as well as the number of methods and classes

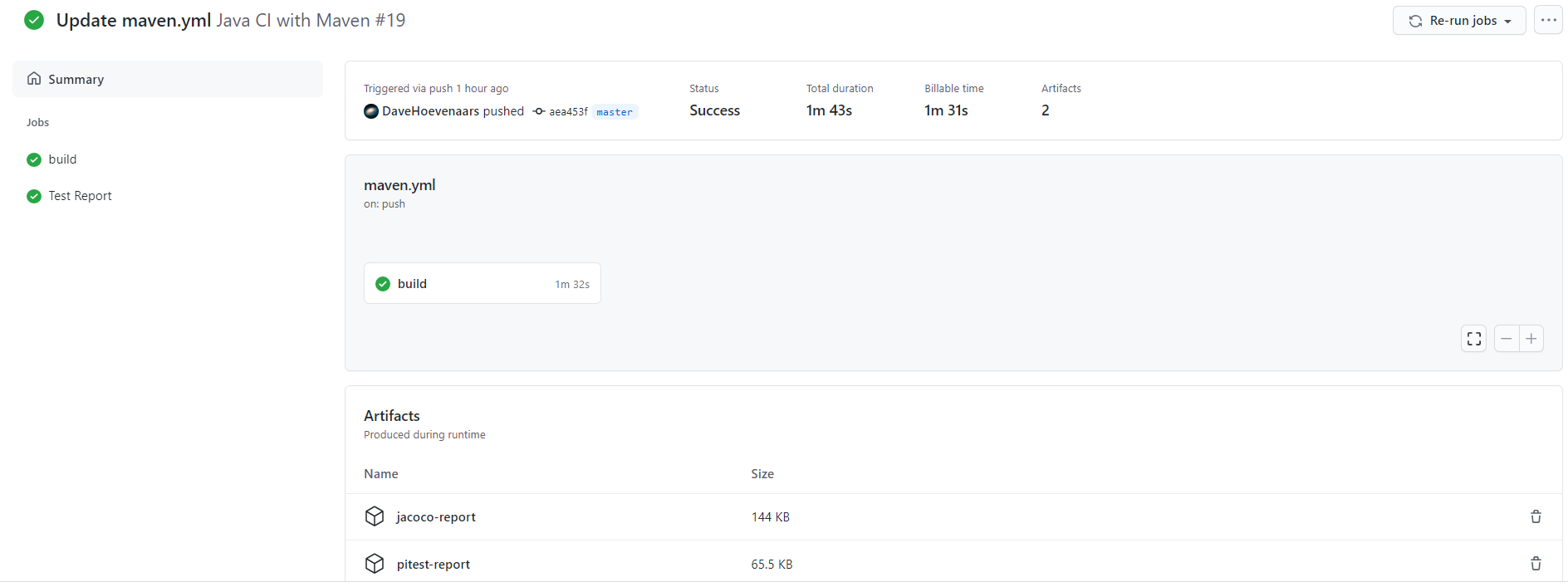


Figure 15 Action job run overview

This is an overview of your action Job run, here you can see which action steps went successfully and which went not successful, in this case they all were successful.



Figure 16 Maven pitest Action

This is the Pitest action defined, not again that you can specify here the target directory

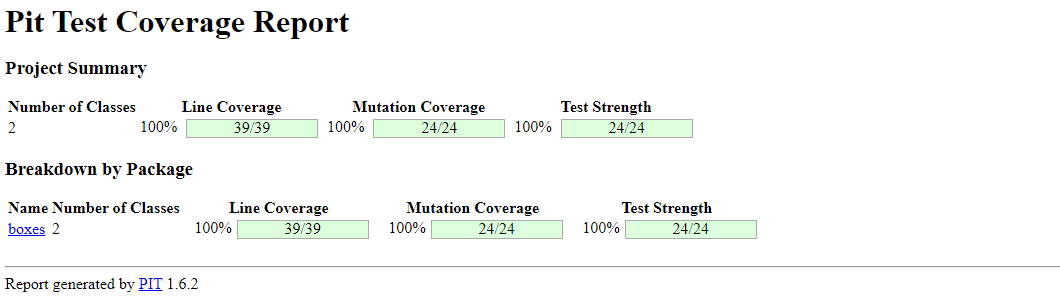


Figure 17 Pit Test report

This is the result of the Pit test ruin, here you can see not only the normal coverage, but also how much changed, and how strong the test are.

# Conclusion & Recommendations

In this section there will be argued why certain automation services would be better for this particular case.

Firstly, I would like to mention that the services provided by GitHub makes a good automation tool for distributing, testing and grading student work. Especially GitHub Actions will provide you with the flexibility and ease of use continuous integration, which as an excellent example for student to follow.

Secondly GitHub classrooms can work alongside GitHub actions really well by taking advantage of the repositories where those actions are defined in, as well as being an effective and automatable Student administration tool.

In this project sequence I managed to construct a prototype which allows for automated software testing and (maven) building. I also was able to make this work in collaboration with GitHub Classrooms. And more importantly, I provided a starting point on which my successors can build on.

Additionally, there is an alternative solution in the works called Codegrade, which aims to also automate testing and grading of the assignments. A drawback for this solution is that Codegrade as a service which is per definition controlled by a third party. Aside from this solution being more costly it would also entail the need for change requests that may cost valuable time if something needs to be changed. Whereas if you would use the GitHub solution you would have this part under your own control. It would also be good that teachers use GitHub extensively so that they understand the technology better and will be able to help the students when they run into problems.

Finally, my recommendation would be to continue along the path of using GitHub Actions and Classrooms in order to provide a stable, modern versioning system(industry standard), which to some degree is able deal with requirement to keep the student solutions private to the student and the teacher solutions secret for all students. Some degree, because, in order to use GitHub Actions on a student project, you will need to define that Action in the template project on which the individual student assignment is based. This means that students would be able to see how these Actions are defined and could see what the teacher tests are. This is undesirable because the teachers want to be able to test the student code, with student code(called BB tests) which the students of course can see. But also with their own defined teacher tests(called AB) tests

For the problem of student to able to see the AB tests, there is a work around. In the Action that you define in your template project, which the student project will make use off, you can link to Actions defined elsewhere and call the tests from a remote repository, which should be private so the student cannot access it. ~~As long as this repository is within your organization~~, you can make a secret to access it, you will then be able to retrieve the artefacts you need and use them for the teacher tests. The student will be able to see that secrets are being used, but will not able to see what the secret is.

Additionally GitHub supports test tampering detection which will notify you when a student has altered the test, however this will only work when your are using GitHub classrooms with autograding turned on.

In the current situation, before test are run the sources of both tests either A or B and business code also either A or B is copied to a RAM disc environment to reduce disk wear and speed up testing. Let’s call this process composing the test environment. After composing the test environment, subversion secrets are no longer needed for the duration of the test and be kept invisible for the test, for instance by running the test in a separate container(Docker), that does not have any network or disc connections. This approach thereby could prevent secret leakage. Version system checkout can be decoupled from running the tests.

# Literature List

* GitHub Guides
* YouTube
* Mozilla
* Peerweb(PRC2 Page)

# Resources

GitHub Actions Tutorial:

<https://docs.github.com/en/actions>

GitHub Actions Demo:

<https://youtu.be/cP0I9w2coGU>

**Summary GitHub Actions:**

You can automate all the actions in a normal github work flow(main, branch) +

But also add custom actions in between those normal actions (Custom software development lifecycle workflows directly in your GitHub repository. +

Github actions are event driven, meaning that you can run a series of commands after a specified event has occurred(like a pull request).

Github actions use YAML syntax to define the events, jobs and steps(this is stored in your repository @ .github/workflows

In this case push is the event that triggered the action, and uses steps to control the order in which actions are run which automate your software testing.

These steps are: GitHub actions checks out the pushed code, install the software dependencies, runs bats-v to output the software version

Furthermore you can build test and publish across multiple Operating systems and languages in one workflow.

Canvas API:

<https://developer.mozilla.org/nl/docs/Web/API/Canvas_API>

GitHub Actions Template Workflow:

<https://github.com/actions/starter-workflows/blob/055373ee0b531de9b779896c520d0555e7df48ae/ci/blank.yml>

GitHub Packages Explained:

<https://www.youtube.com/watch?v=N_-Cu9_2YAA>

**Summary GitHub Packages:**

Enabling sharing of packages that are to be trusted and that you can rely on

Fully integrated with github

Anyone with an account can publish registries

Works with npm, maven, docker, ruby registry protocols.

Unless you have a personal access token, you get not directly get packages available from github

ACCESS TOKENS can be given certain permissions.

GitHub Classroom Setup Tutorial:

<https://www.youtube.com/watch?v=KXWXg68KpTY>

Translating Test results to Canvas or other student administration systems:

<https://youtu.be/KXWXg68KpTY?t=702>

Creating Maven packages with GitHub:

<https://www.youtube.com/watch?v=MhzoxE7NdpI>

How to deploy a Maven Package:

<https://docs.github.com/en/actions/guides/building-and-testing-java-with-maven>

Sure Fire report Template:

<https://github.com/marketplace/actions/surefire-report>

JaCoCo Coverage:

<https://github.com/marketplace/actions/jacoco-report>

JaCoCo Badge:

<https://github.com/marketplace/actions/jacoco-badge-generator>

Conditional PIT Test with PR-Landmine:

<https://github.com/marketplace/actions/pr-landmine>

PRC2 Fontys way of testing during Performance Assessments

<https://prc2.fontysvenlo.org/2021/assessmentcorrectionrules.html>

GitHub Secrets:

<https://youtu.be/3bz0IR-GDIw>

Pitest:

https://pitest.org

# Questions

1. Can GitHub do functionally the same as the current systems in place? With some adaptations, yes
2. If students have been imported from Canvas to GitHub Classrooms, can they still be added? Yes through a CSV file, if your GitHub Classroom is linked to a Canvas systems you’ll have to add the students there.
3. Can students access the GitHub actions? If they are defined in the assignment folder then yes, there is a solution in the work however in which the actions are called from a private repo in which cases the students won’t have access.
4. How to publish JaCoCo Coverage? Through Canvas or Email.
5. Where to the Maven build artefacts end up? In the target folder of your repository

1. Definition by PRC2 Fontys way of doing performance assessments [↑](#footnote-ref-1)
2. Definition by PRC2 Fontys way of doing performance assessments [↑](#footnote-ref-2)