Università
della
Svizzera
italiana

Faculty
of Informatics

Bachelor Project

June 16, 2018

Online platform for Productive Failure

Susanna Riccardi

Abstract

For my bachelor project I implemented an **online platform** that implements the Productive Failure approach, which is a way of designing teaching so that it addresses failure, instead of waiting for it to happen.
The reason I have worked on developing an online platform for this approach is to improve its **scalability**: as of now, it is being applied in classrooms of 20-40 students by teachers using paper and pen. Having an online platform means that students all over the world can use this approach without necessarily having to be in a class.
Moreover, the project has been designed to be used for teaching concepts of Computer Science. Due to time constraints, the project has been developed to teach only one concept, specifically the concept of **iteration**; however, one of the goals of this bachelor project was to design a platform so that it could be easily expanded to teach other concepts in the future.

Advisor
Prof. Matthias Hauswirth

Contents

1 Introduction

When we think about education, we have the idea of a teacher explaining concepts in front of class, which is silently listening, taking notes, and sometimes asking questions. In the last few years, there has been some research in this field, which has brought to light the concept of **Productive Failure**. Analyzing the name can kind of tell us about what is it about, which is designing the way of teaching so that it addresses failure, instead of waiting for it to happen.

A practical example can help make this more clear. Let's imagine a teacher who wants to explain the concept of standard deviation to a classroom of 20 students. Instead of directly explaining it to the students, he/she provides them with a **problem**: given a data set of two football strikers, students are asked to invent as many different ways of mathematically deciding the more consistent player[1]. This problem is designed so that it could be solved by using the standard deviation, but the students don't know yet about it: they need to use and apply their knowledge to come up with their own solutions to it. When they are done, the teacher analyzes in front of the classroom all the student-generated solutions, compare them and contrast them, to come up with a *general solution* that applies the standard deviation.

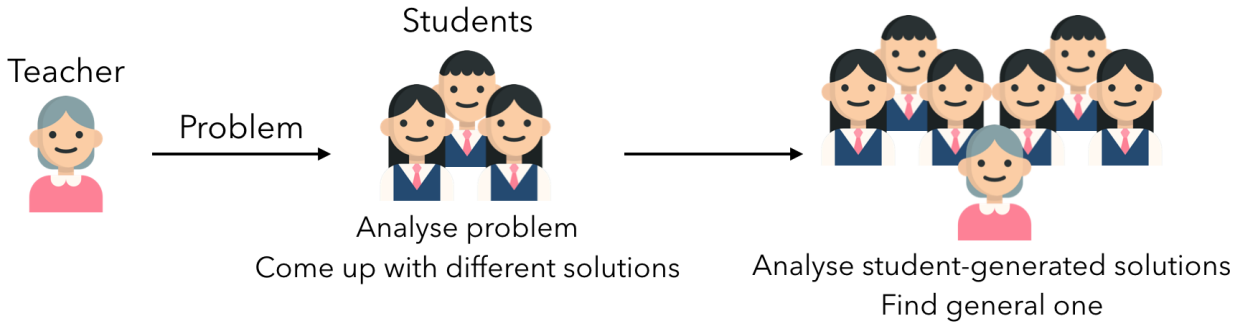

Figure 1. Illustration of Productive Failure approach [2]

As of now, the Productive Failure approach has been implemented (and tested) only in classrooms of 20 to 40 people, using a paper and pen way: there has been no testing in a much bigger context. The goal of this bachelor project is to **scale this approach to a bigger audience**, so that, in the future, its benefits can be tested. In order to do so, I have developed a web application that people from all over the world can use to learn Computer Science concepts.

One target of this project has been to *design an online activity* that would follow Productive Failure. I decided to go for a **teacher-free implementation**, so that students from all over the world can take part in it without the need of a teacher to be present.

Due to the time limitations of the project, I designed the activity to be about only one concept, specifically the concept of **Iteration** in the Computer Science field. However, one objective has been to design the platform so that it could be expanded to teaching other concepts.

The activity is composed of *two phases*: the first one is called **generation**, which is divided into two steps: in the first step, students have to analyze two patterns and come up with different representations for them. A **representation** is a description of the pattern, that can be composed of text, shapes and arrows. The idea is that the student should try to describe the repetition in the pattern without describing it literally (that is, not writing/drawing the provided pattern in its whole). In the second one, they see some representations that were previously generated by other students, and they have to reproduce the original pattern. In the second phase, called

consolidation, students are asked to rank some representations, and they have to understand why some of them are better than other ones.

The remainder of this report is going to be structured in the following way: first, I give a description of the Productive Failure approach as a background; then, I explain the approach I have taken to build the platform. After that, I illustrate the implementation, which tackles the more technical side of the project; lastly, I give a description of the result.

2 Productive Failure

In the recent years, a lot of research in the field of education has been made. Specifically, with regards to failure in learning, a new concept has been developed, which states that failure has an impact on learning. This concept is called **Productive Failure**[1]: it is about designing the way of teaching so that it addresses failure, instead of waiting for it to happen, and it is based on the belief that engaging a person to try - and even fail - at tasks that are beyond its skills and abilities can, under certain conditions, be productive for developing deeper understandings of the concepts that are thought.

2.1 Designing for Productive Failure

When designing for this approach of teaching, a key thing to keep in mind is that students should be working on these features:

- activation and differentiation of *prior knowledge* in relation to the targeted concepts;
- attention to critical conceptual features of the targeted concepts;
- *explanation* and *elaboration* of these features;
- *organization* and assembly of the critical conceptual features into the targeted concepts.

The analysis of these concepts has brought the Productive Failure approach to be composed of two phases: the first one is the **generation phase** where the students have the opportunity of generating and explaining the affordances and constraints of multiple solutions to given complex problems. In the second one, called **consolidation phase**, the students, alongside their teachers, have to compare and contrast, organize and assemble the relevant student-generated solutions into canonical solutions.

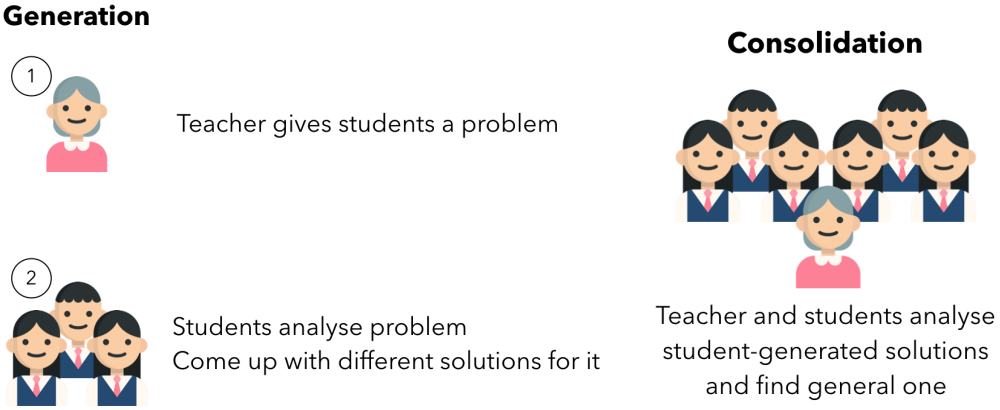

Figure 2. Illustration of Generation and Consolidation phases

While designing for **Generation phase**, it should be noted that:

- students should be challenged, but not frustrated, and they should remain engaged with the problem throughout the phase: this concerns the creation of problems, with different developments for different levels of education;
- the problems should have a variety of different solutions and they should be solved without prior knowledge.

2.2 Evidence

The Productive Failure approach has been tested and replicated in both controlled-experimental and quasi-experimental studies in laboratories and classrooms with real students and teachers. These are the **evidence** that came up during those studies:

- Productive Failure students performed better on *conceptual understanding* and transferring to novel situations, without compromising basic procedural fluency in the domain;
- the number of solutions that students generated prior to instruction significantly predicts how much they learned from the instruction;
- students with *different level of abilities* were able to learn just as well from Productive Failure.

For more information about the evidence for Productive Failure, see [1] and [3].

2.3 Improving the approach

The Productive Failure approach has been applied in classrooms composed of 20 to 40 students, only using paper and pen. The benefits of this are shown in the previous section. However, the approach has not yet been studied and applied to large-scale settings. Also, since Productive Failure is not designed for the individual learner, work has still to be done in this field. Specifically, the two main tasks the researchers wrote about are:

- Improving **efficiency** and **effectiveness** of Productive Failure, by *taking it to scale*: applying this approach to a classroom implies that the teacher must have a background of the concepts they are trying to teach. By implementing an online platform that can be used for Productive Failure, this approach could be used to teach, for example, programming in schools where teachers don't have a background in Computer Science.
- Understanding and modeling how Productive Failure works: while it is clear and proved that this approach brings benefits to teaching, it has not yet been proven *why* it does.

This project focuses on working on the first task, specifically on the topic of Computer Science. This platform could then be used for the second task. This different way of teaching has already been applied to Math. For this project, it will be applied to Computer Science.

3 Approach

One major task of this project was **designing the activity** presented on the platform. The way the Productive Failure approach is implemented currently is the following:

- first, the classroom is divided into groups, and each one of them is presented with a problem; the students have to analyze it and come up with as many solutions as they can;
- then, the teacher, alongside the students, analyzes, compares and contrasts the generated solutions, and finds a more general one.

As you can see, there is a *high dependence on a person* (the teacher) that needs to wrap up the activity and it has to be conducted by different people at the same time. These two things were making the approach too dependent on a number of different people, all at the same time. However, when thinking about the scalability of a project, the idea is that it should be taken on by as many people as possible, but there shouldn't be a dependency between them.

3.1 Deciding how to structure the platform

The solution that I decided to implement was a middle ground between two possible implementations: one in which the user is completely independent, where there is no teacher and no contact between students, and one that takes into account the need for a teacher. The activity is **peer-based**: this means that once the students provide their solutions, they are presented with other students' one, that they have to analyze. This makes the activity completely *teacher-independent*, but there is still some (indirect) contact with other people.

Using this approach means that we needed to think on how to make it respecting the Productive Failure way. The final result is a three step activity: the *generation phase* is divided into two steps, one in which the students provide solutions, and one in which they are provided with solutions generated by other students; the *consolidation phase* is a one-step activity in which the students analyze teacher-generated solutions to consolidate their knowledge.

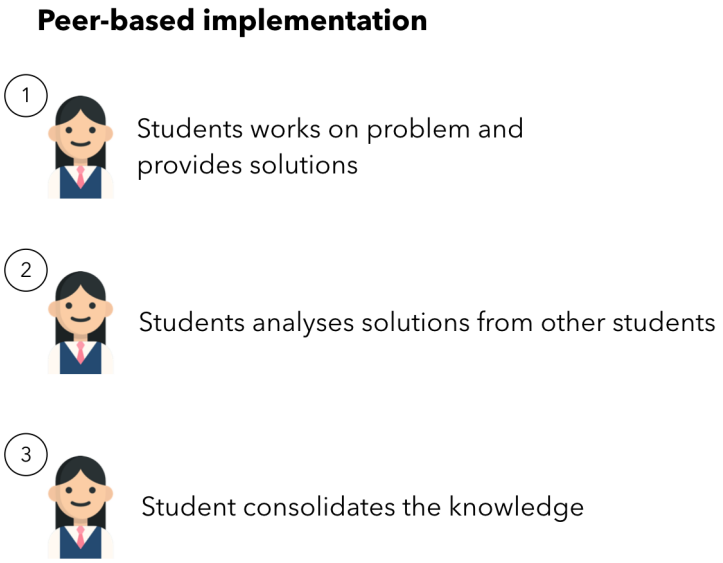

Figure 3. Illustration of the approach

3.2 Small pilot study

At the beginning of the project, Prof. Matthias Hauswirth proposed a Productive Failure activity during one lecture of his Programming Fundamentals 2 course. He divided the classroom in groups of four-five people, and he assigned four identical papers to each one of them: on each one, there were **two patterns**.

The **goal** of the activity was for the students to *identify the repetition* in the patterns, and come up with solutions to describe these repetitions. Moreover, for each representation they draw, they had one **constraint**, for example not using text, arrows, pictures, or drawing the shortest description possible. As you can probably imagine, the activity was directed towards teaching the concept of Iteration.

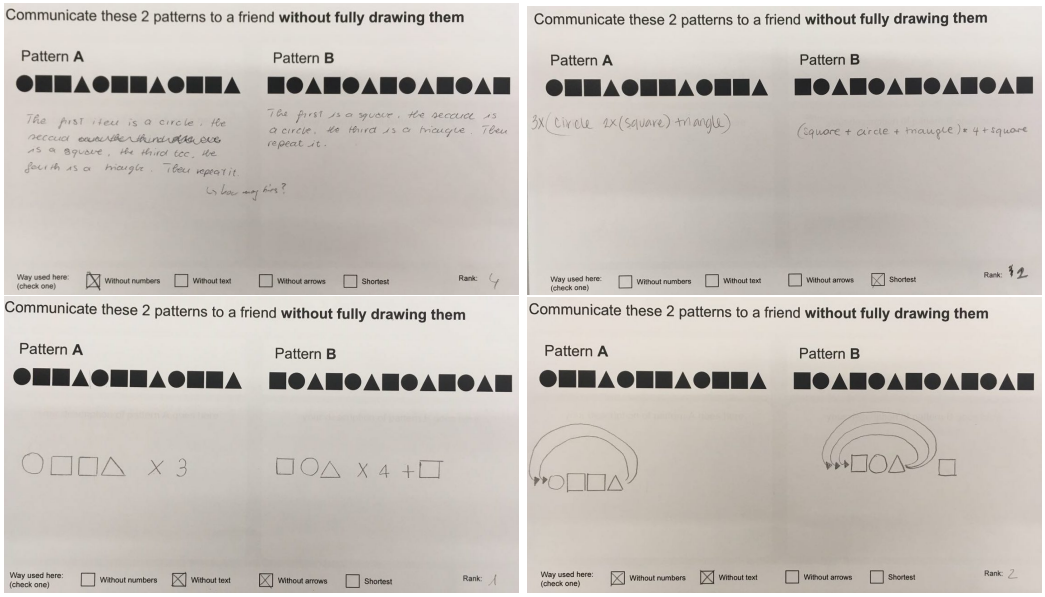

Figure 4. Some representations collected during the experiment

The activity was really helpful for planning my project: I got some sample representations from real students, I received some feedback on how the user interface should have been, and also I managed to design the structure of the activity.

3.3 Why iteration

Due to *time limitations* on the project, which lasted about 14 weeks, there were some discussions on what to focus during the development of the platform. Professor Hauswirth and I decided to concentrate on the implementation of an activity that was designed to teach iteration, mainly because of the sample activity that he did in class, described in the previous section. In this way, at the end of the project I would have created something that could be used, but also expanded with a whole new variety of concepts.

3.4 Structure of the platform

After talking with Matthias Hauswirth, Robert West and Manu Kapur, we came up with the following structure for the platform:

- the student logs in to the platform, by creating an account;

- the user can start a *Productive Failure activity* on the concept of Iteration, which is divided into three parts: the first two steps are consistent with the generation phase of PF, the last one is consistent with the consolidation phase.

In the section 5 I will describe more in details the activity I created.

4 Implementation

The decision of the technologies was an important one, because we needed to keep in mind that the project was supposedly going to be used by a lot of people. One option that we took into consideration was to integrate our application with platforms that are developed for teaching Computer Science and that already have the infrastructures for such an activity.
Among all the different possibilities, we decided to take a closer look at **Code.org**.

4.1 Code.org

Code.org[4] is a non-profit organization that has the goal of encouraging people to join the Computer Science community, by learning Computer Science with online courses hosted on their website, namely *Code.org*. The website is opensourced on *GitHub*[5], and people can actively contribute to it.

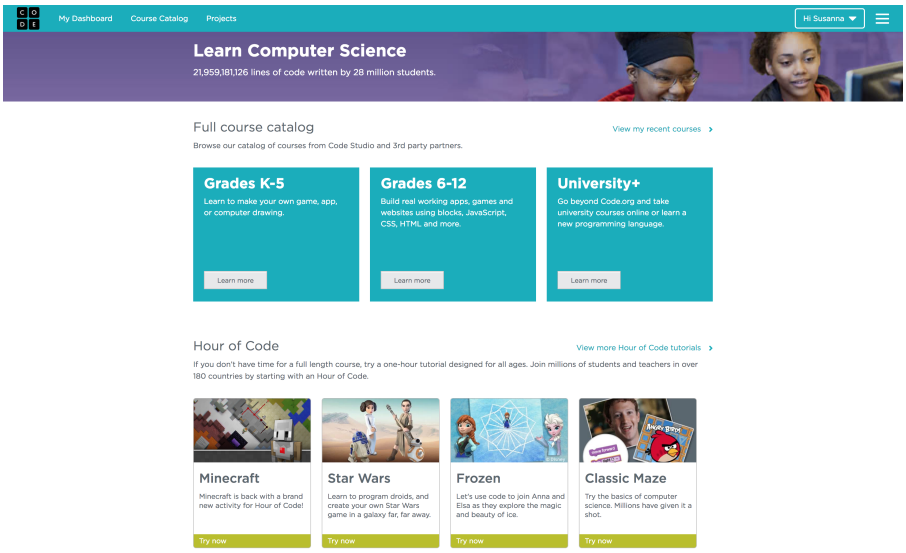

Figure 5. Code.org main page

The website offers three curricula:

- *Computer Science Fundamentals*, oriented to students of Elementary schools;
- *Computer Science Discoveries*, oriented to Middle school students;
- *Computer Science Principles*, oriented to High school students.

These courses can be joined by students from all over the world. Students can create their own account and start as many activities as they want. The website offers two kind of **accounts**: one for *students*, which gives access to all the courses and activities, and one for *teachers*. Moreover,

the website offers **Free-to-play projects**, which the students can use as a template, to let their imagination flow. Some of them are: Artist, Frozen, Play Lab, App Lab, Game Lab, Web Lab.

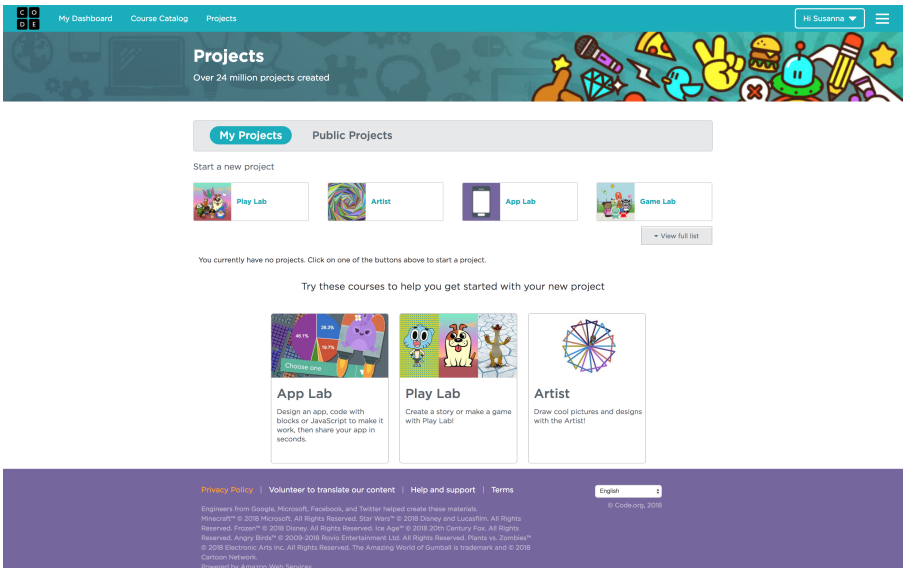

Figure 6. Code.org projects page

The codebase of Code.org is structured in the following way:

- The backend is a *Ruby on Rails*[6] application, which is responsible for the courses and tutorials, the user accounts and the students progress;
- The frontend is written in *JavaScript*. Specifically, some Free-to-play activities are written using React[7], which is a JavaScript framework for constructing single page web applications.

The main reason we decided to go with Code.org is that their codebase is *opensource*, and we could have worked on it directly. Our activity would have been developed like App Lab or Game Lab, which means that it would have been self standing. After evaluating all the pros and cons of developing the Productive Failure activity within the Code.org repository, we decided, due to the time limitations, it was best to work separately, but leave the opportunity for a merge in the future.

4.2 Technologies

When evaluating Code.org, I spent some time trying to understand its codebase. This included spending time on learning the technologies that were behind it. As mentioned before, the Backend was written in *Ruby on Rails*, and the Frontend was written in *React*. Since I already had previous experience with React, I only had to spend some time learning Ruby on Rails. I used online tutorials on Codeacademy to get an introduction into this language and platform.

As mentioned before, we wanted to have the opportunity in the future to merge our activity into Code.org, which is one of the reasons why we decided to have the same technologies as them: our platform has a server written in Ruby on Rails, and the Frontend is written in React. Another reason we decided to go with React for the Frontend is that the way it is structured makes it easy to reuse components for different activities.

After evaluating all the possible options[8], I decided to:

- create a server in Ruby on Rails that implements a REST API;
- create a React app that runs locally with npm, and that can request data to the REST API.

The next two sections are going to explain in detail how the Backend and the Frontend communicate.

4.2.1 Backend

The Ruby on Rails server implements a REST API, to which the React app performs AJAX requests. The server starts with the command *rails s -p 3001*. The template I used has a SQLite3 database. In the folder *'/app/controllers/api/iteration'* there are all the controllers that are used for retrieving and saving data in the database, and serving it to the different routes. The API can be expanded by adding subfolders in the *api* folder. The file *routes.rb* in the folder *config* contains all the available routes: all the routes are under the */api* namespace. The activity for iteration is under */api/iteration*. When adding a new activity, just add a new namespace under the */api* namespace. The file *schema.rb* contains all the information about the tables in the SQLite3 database.

4.2.2 Frontend

The Frontend was generated using *create-react-app*: this generated a template that I used to create the online platform for Productive Failure. Running *npm run start* runs the app in the development configuration. *npm run build* is used to compile the project, which can then be deployed by serving it statically.

The folder contains the following subfolders:

- *public* that contains all the static content that is loaded at the beginning, like *index.html*.
- *src* consists of all the JavaScript files. This contains index.js, which is the first JavaScript file that is loaded, App.css, which contains all the styling for the website, and two other folders: *img*, which contains all the images for the website, and *components*, which contains all the React components about the different (possible) activities on the application. Inside this last folder there are subfolders used for organizing the different components. For example, there is the *iteration* folder, that contains all the components related to the created activity.

5 Results

As mentioned in the previous sections, the users can log in to the platform after creating an account, and they are presented with the Productive Failure activity about **iteration**. The activity is composed of three steps:

1. the first one is the first step of the Generation phase, and it is called **Step one: identify and describe patterns**;

2. the second is the second step of the Generation phase, and it is called **Step two: interpret pattern descriptions**.

3. the last one is the Consolidation phase, but it is referred to as **Step three: combine insights from different solutions**.

Each step is presented with a tutorial, in which there is a description of what has to be done. Below you can find a description of the activity.

5.1 Step one: Identify and Describe Patterns

When first loading the page, students see a tutorial pop up, that explains the goal of this step of the activity. The tutorial can be accessed whenever the students want during the activity by clicking the **Get info about this step** button. The tutorial is composed of two steps: the first one explains the general goal of the activity, and the second one tells about the goal of the current step and what to do.

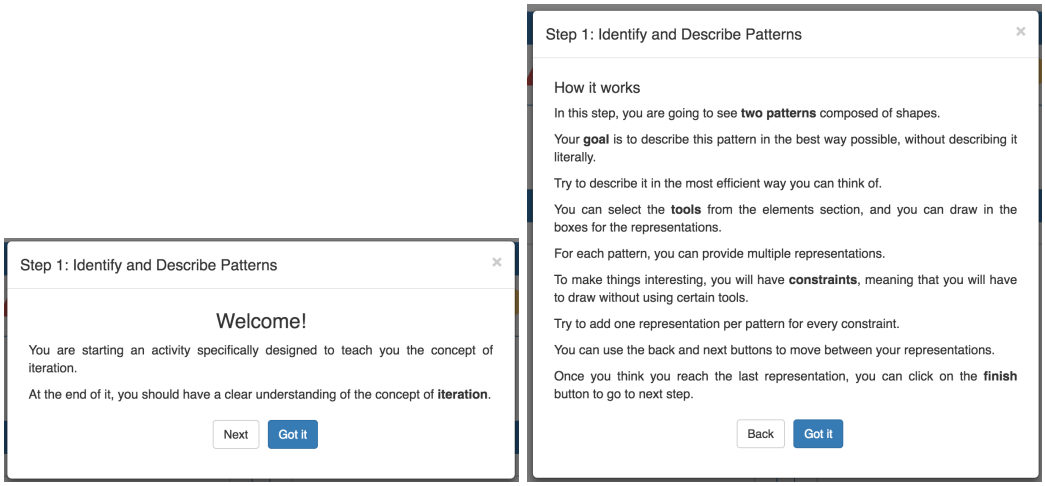

Figure 7. Tutorial for the first step

In the first part of the generation phase, users are presented with two *patterns*, along with two canvases that they can use to write and draw. In this section, the user has the goal to identify the *repetition in the two patterns*, and provide representations of them; also, they need to describe the patterns in their own way, without describing them literally. This means that they have to think about the patterns, analyze them, and come up with **representations** that take into consideration the repetitions in the patterns.

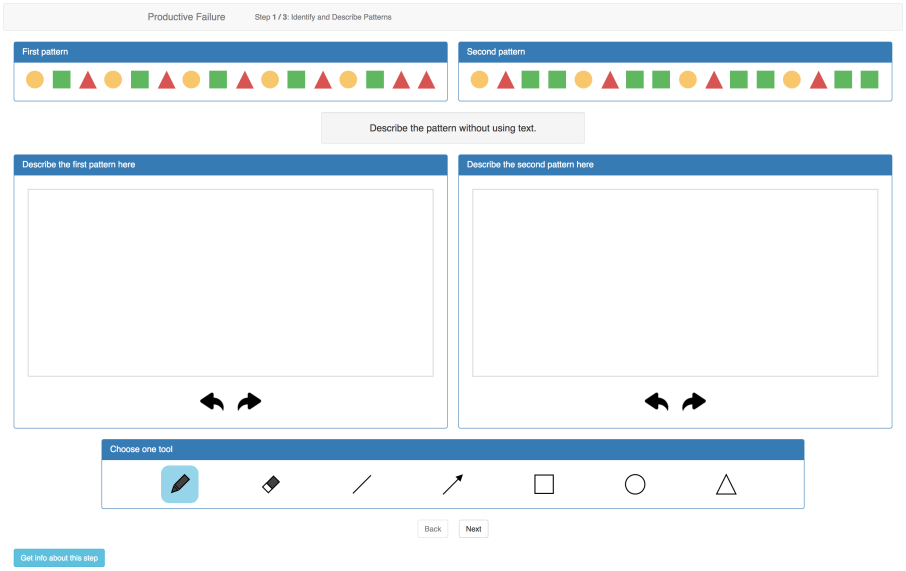

Figure 8. User interface for this step

Moreover, to make things a little bit more interesting, and to try to stimulate the users' creativity, some **constraints** are provided, meaning that the users can't use certain tools while drawing a representation. For example, one constraint could be that the user can't write text. The goal for this is to have the users try to reason about the patterns in different ways, knowing that they must come up with at least 4 different representations (assuming we have 2 constraints). The users can go back and fourth between their representations, and modify them as many times as they want. When they are ready to go to the next phase, they can submit their representations.

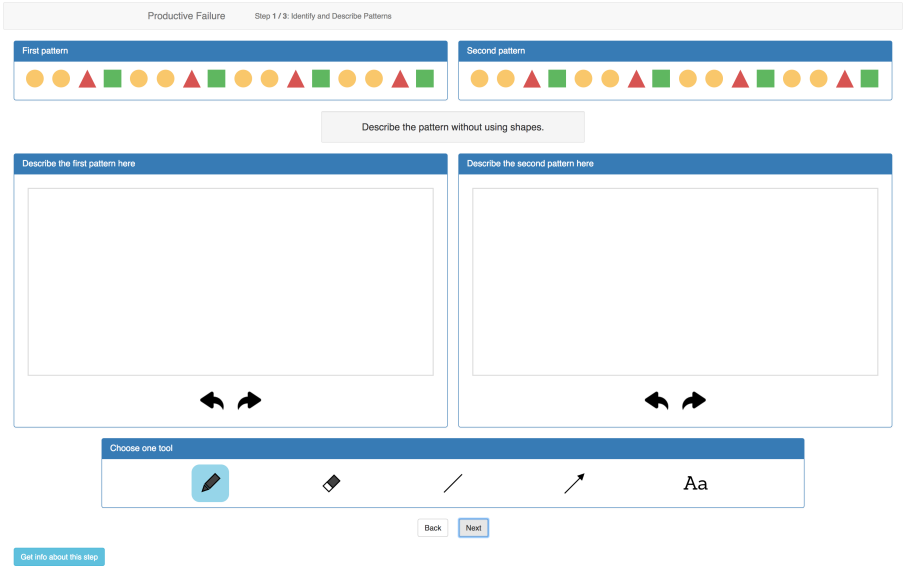

Figure 9. User interface with another constraint

The reason why we present two patterns to the students is for having two contrasting cases of the same thing: by contrasting cases, I mean that there is a clear difference between the two. As you can see in Figure 9, in the pattern on the right there is a repetition, but also a suffix. In the pattern on the right, it's only repetition.[9]

As mentioned previously, the student can provide multiple representations: this is because of the idea behind Productive Failure, which is that the students have to reason about a problem, and think about it from different angles. If a student is provided, for example, with the constraint of not using text, he/she will have to try to solve the problem in a different way.

5.2 Step two: Interpret Pattern Description

Again, when first loading the page, students see a tutorial pop up, that explains the goal of this step of the activity. The tutorial can be accessed whenever the students wants during the activity by clicking the **Get info about this step** button.

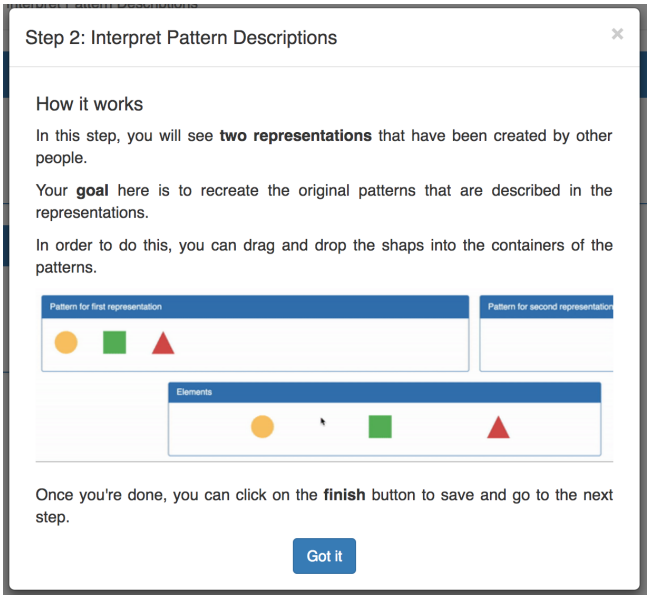

Figure 10. Tutorial

Once the users are taken to the second part of the generation phase, they are presented with two **representations** that were previously generated by other students. The **goal** of this step is to analyze the representations, and recreate the patterns that were first given to the students who created those representations. The users are provided with two containers in which they can drag and drop shapes to recreate the pattern. Once they think they are done, they can submit their patterns and go to the last step.

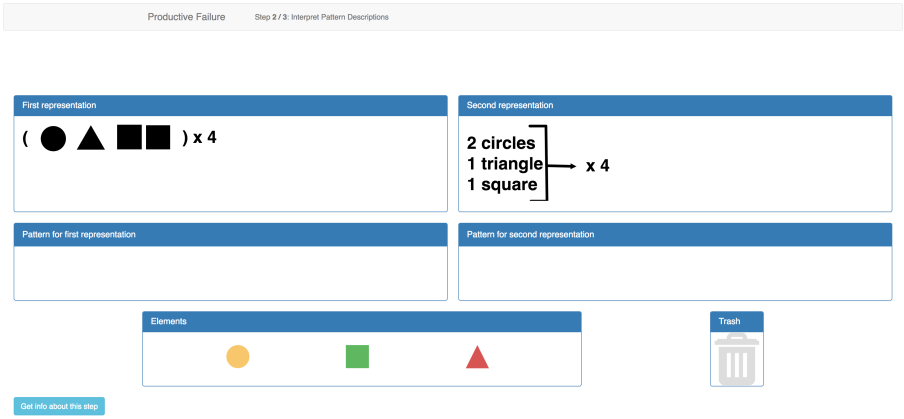

Figure 11. User interface

As mentioned in the *Approach* section, the generation phase needed to be restructured to be made teacher-independent and without the need for contact with other people. This is why we decided to divide the generation phase into two steps. The goal of this step is to have the student analyze some representations that were previously generated by other students: this is done so that we can automatically detect whether a representation is good: if other students are able to reproduce the pattern from the representation, it must be good.

5.3 Step three: Combine Insights from Different Solutions

As in the previous steps, when first loading the page, students see a tutorial pop up, that explains the goal of this step of the activity. The tutorial can be accessed whenever the students wants during the activity by clicking the **Get info about this step** button.

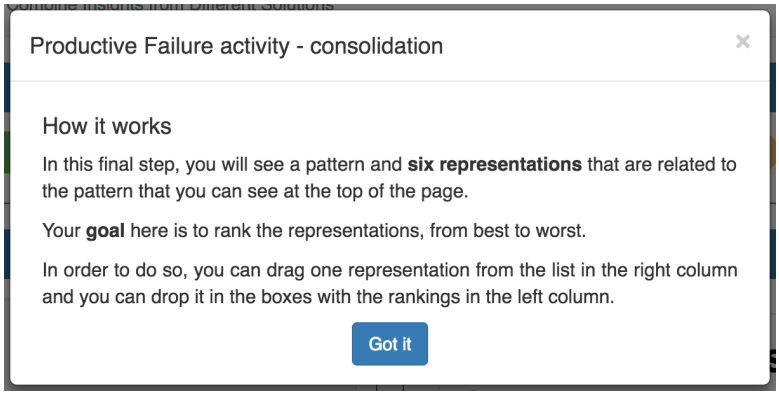

Figure 12. Tutorial

The last step of this activity is *consolidation*: here students are presented with a pattern and six corresponding representations, and the **goal** is for them to rank these from best to worst.

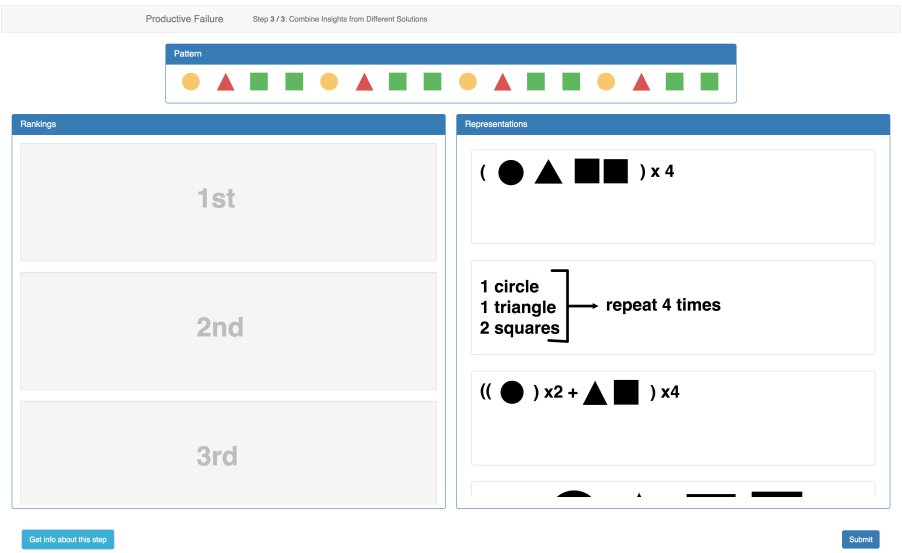

Figure 13. User interface

Once they do this, for each representation a description is shown: this describes the representation, its pros and cons.

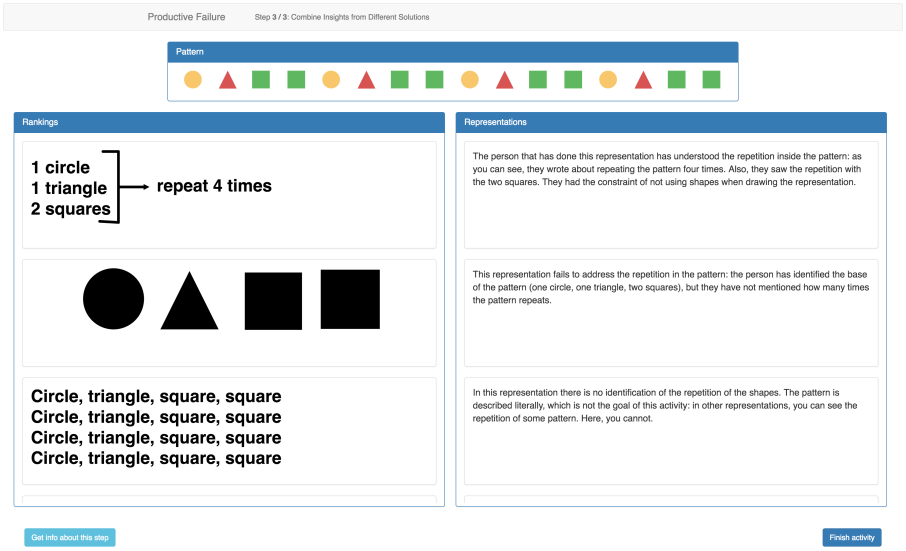

Figure 14. Tutorial

The goal of this step is to make the students activate the knowledge from the two previous steps to verify if they understand why each representation is good or bad.

6 Future implementation

The current project can be expanded in multiple ways. First, some work can be made on making the activity more complex: as of now, it is about basic iteration, showing simple one-dimensional patterns. Two-dimensional patterns could be added, making the activity expand the concept of iteration to nested loops.

Another future implementation could be to implement **other activities** about Computer Science topics. As mentioned in this report, for time limitations the activity was designed for iteration. However, it is prone to modifications and expansions, and could be used to teach multiple other concepts, like recursion, Boolean logic, and so on.

Another addition that could be made is adding a **grading system**, which means that, after completing the activity, the user is presented with a score of his/her performance.

Also, the data that is being collected (representations, correctness of patterns...) could be analyzed to have some statistics about the activity.

7 Conclusion

I am satisfied with how the project turned up. The application is ready to be deployed and used right now, but the ultimate goal would be to integrate it into the codebase of **Code.org**: this would help us reduce the costs for infrastructures and, most of all, it would make our activity reachable for thousands of students all over the world.

The students can log in into the platform, and take on an activity on Iteration. The activity is composed of three steps. In the first step, the students see two patterns, and have to recognize the repetition in them. They have to describe the repetition inside the canvases shown in the page, and they have to respect some constraints that are provided.

In the second step, the students are shown two representations created by other students during Productive Failures activities, and the goal is to reconstruct the original pattern presented

to them.

The last step has the goal to let the students consolidate what they have learned during the activity. This is done by having them rank six representations generated by a teacher, and show them a description of why and how it is good or bad.

The project is open to future improvements. Also, it would be nice to test the benefits of this platform in the context of Productive Failure.

References

[1] M. Hauswirth R.West, M. Kapur. Digital, personalized, and adaptive learning from productive failure. Unpublished grant proposal.

[2] Images by Freepik, from Flaticon. School pack, 2016. https://www.flaticon.com/packs/education-11.

[3] K. Bielaczyc M. Kapur. Designing for productive failure. *Journal of the Learning Sciences*, 21(1):45–83, 2012. https://doi.org/10.1080/10508406.2011.591717.

[4] Code.org. Website. https://www.code.org.

[5] Code.org. Github repository. https://github.com/code-dot-org/code-dot-org.

[6] Ruby On Rails. https://rubyonrails.org/.

[7] React. https://reactjs.org/.

[8] Hrishi Mittal. How to build a react app that works with a rails 5.1 api, 2017. https://reactjs.org/.

[9] M. A. Oppezzo D. B. Chin D. L. Schwartz, C. C. Chase. Practicing versus inventing with contrasting cases: the effects of telling first on learning and transfer. *Journal of Educational Psychology*. https://aaalab.stanford.edu/assets/papers/2011/Practicing_versus_inventing.pdf.