UNIVERSIDAD
REY JUAN CARLOS

MÁSTER EN INGENIERÍA DE TELECOMUNICACIÓN

Curso Académico 2019/2020

Trabajo Fin de Máster

ANALYSIS OF BAD SMELLS IN PROGRAMMING WITH DR. SCRATCH

Autor : Ángela Vargas Alba

Tutor : Dr. Gregorio Robles

Trabajo Fin de Máster

Analysis of Bad Smells in Programming with Dr. Scratch

Autor : Ángela Vargas Alba

Tutor : Dr. Gregorio Robles Martínez

La defensa del presente Proyecto Fin de Máster se realizó el día de Marzo de 2020, siendo calificada por el siguiente tribunal:

Presidente:

Secretario:

Vocal:

y habiendo obtenido la siguiente calificación:

Calificación:

Fuenlabrada, a de Marzo de 2020

Dedicated to

my family, partner and friends.

II

Acknowledgements

With this work, I finalize one of the most beautiful and difficult periods of my life, my stage as a student. I have achieved this great challenge, first of all, thanks to my family, my parents and my brother. They have supported me since I was a child, helping me in all my decisions and doing all their efforts unconditionally. Thanks to them today I am an engineer.

Thanks to my partner, always by my side. He has accompanied me during all these years, being my best support, colleague, friend and love. I could not have chosen better, I love you.

Thanks to my friends, my girls since we were three years old. We have grown together, living and enjoying each stage of our lives. A friend is a treasure, so I am the luckiest person in the world with you.

I also want to thank my schoolmates during the degree. We started by helping each other in the battle and, at the end, you have become my friends. Thanks for helping me in the worst moments.

I would like to thank all the Dr. Scratch team. Especially, thanks to Cristian, who helped me during these two years and taught me so many things.

Finally, I could not have developed this project without my tutor, Gregorio. Thanks for this great opportunity and for all the things that I have learned from it.

To all of you: Thank you.

Summary

This project consists of the evaluation and analysis of the impact which bad habits have in the programming area. In particular, in this work we want to analyze the effect of "bad smells" in the development of the Computational Thinking skills.

For its development, we have based on the Dr. Scratch tool, a free software web application which allows the analysis of projects designed with Scratch -programming language oriented to education- and to obtain an assessment about different aspects related to the Computational Thinking.

The final objective of the project is the implementation of a new assessment model in Dr. Scratch which allows to raise awareness and prevent about the use of "bad smells" in programming with Scratch.

In order to carry out this process, several phases of work have been necessary, with different technologies involved. In an initial phase, it was needed an update of the Dr. Scratch tool. The technologies used for that were related to web programming and the cloud production environment, such as Django, MySQL, Microsoft Azure or Google Cloud Platform, among others. During the second phase, we carried out an exhaustive analysis about "bad smells". For this procedure, we used Jupyter Notebook, an appropriate technology for the data analysis. For the web design of the new model described previously, which constitutes the third phase, technologies such as HTML, CSS and Bootstrap were used. Finally, in order to verify the effectiveness of the project, we designed and implemented an assessment experiment with different teachers. To achieve this last phase, Google Forms together with the Dr. Scratch tool were used.

Resumen

Este proyecto consiste en una evaluación y análisis del impacto que tienen los malos hábitos en el mundo de la programación. En concreto, trata de analizar el efecto que producen los "bad smells" en el desarollo de las habilidades del Pensamiento Computacional.

Para su desarrollo, nos hemos basado en la herramienta Dr. Scratch, una aplicación web de software libre que permite analizar proyectos diseñados con Scratch -lenguaje de programación orientado a la educación- y obtener una evaluación sobre diferentes aspectos relacionados con el Pensamiento Computacional.

El objetivo final del proyecto es la implementación de un nuevo modelo de evaluación en Dr. Scratch que permita concienciar y prevenir sobre el uso de "bad smells" en la programación de proyectos Scratch.

Para llevar a cabo este proceso, han sido necesarias diversas fases de trabajo con diferentes tecnologías implicadas. En una fase inicial, fue necesaria una actualización de la herramienta Dr. Scratch. Para ello se utilizaron tecnologías relacionadas con la programación web y el entorno de producción en la nube, tales como Django, MySQL, Microsoft Azure o Google Cloud Platform. Durante la segunda fase del proyecto se llevó a cabo un análisis exhaustivo sobre "bad smells'. Para este procedimiento se utilizó Jupyter Notebook, una tecnología propia del análisis de datos. Para el diseño web del nuevo modelo descrito anteriormente, lo que constituye la tercera fase, se utilizaron tecnologías como HTML, CSS y Bootstrap. Finalmente, para comprobar la efectivad del proyecto, se diseñó e implementó un experimento de evaluación con diferentes profesores. Para la consecución de esta última fase, se utilizó Google Forms junto con la propia herramienta Dr. Scratch.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Computational thinking (CT) is a knowledge field of great relevance and interest nowadays. During the last decade, CT has become more significant in sectors as important as education. However, to get an exact definition about CT is an actual challenge. In the year 2006, Wing defined CT as a process of formulation and resolution of problems which employs the fundamental concepts of computing [Wing, 2006]. Although its skills can be developed in different ways, one of the most common tools to learn it, train it and develop it, is through programming. In this chapter, we describe the importance of an appropriate development of CT skills and how Scratch can be a fundamental tool for it.

1.1 General context

New technologies represent a fundamental role in the daily life of children and teenagers. In addition, the fact that technologies are still growing, developing and becoming more important and necessary, is a certainty. In this new era, programming is an essential skill.

Learning how to program since childhood is like learning a new language, the earlier children start, the easier it will be for them to acquire its skills and abilities.

Therefore, when we talk about programming, the intention is not that children learn advanced concepts since childhood. The main purpose in these early ages is that they participate in the digital world in a secure, responsible and conscious way. In this way, new generations will be able to understand the new technologies and use them to solve problems in their quotidian life. The main objective of teaching programming in the classrooms is that students obtain the

necessary tools to manage themselves in a technological world [Mangifesta and Feldfeber, 2019].

The inclusion of programming in the educational field allows the development of CT. CT is composed of different skills which are very necessary for children, such as abstraction, logic or problem decomposition. In this way, CT should be considered an ability as important as the reading, writing or mathematics in the schools [Calao et al., 2015].

However, from a software engineering point of view, we know that problems solved through programming may have not been solved in the most appropriate way. These symptoms or bad practices are known as "bad smells". In other words, the program may run and may even solve the problem, but it contains elements that make it difficult to understand, to modify and to reuse [Zhang et al., 2011]. Martin defines code smells as follows: "Code smells are usually not bugs; they are not technically incorrect and do not prevent the program from functioning. Instead, they indicate weaknesses in design that may slow down development or increase the risk of bugs or failures in the future" [Martin, 2009]. Despite the negative effect they produce, bad smells have been little investigated and analyzed in CT research. As Hermans and Aivaloglou have found in an experiment with Scratch learners [Hermans and Aivaloglou, 2016], we argue that bad smells hinder the proper development of CT skills in learners. Their identification should be a first step to guiding learners towards good practices which offer them the possibility to develop themselves to their full potential.

Years ago, learning to program was a complicated task. The information to start this process was scarcer and the programming languages were less intuitive. Programming based on best practices was even more complicated. Nowadays, there are numerous tools and languages to begin in the world of programming and CT. In the following section, we describe Scratch, a visual language of programming designed for children and beginners, which is already used by millions of students worldwide.

1.2 Scratch

Scratch[1] is a visual programming language oriented to education. It has been designed by the MIT to facilitate the learning in an intuitive way, through the use of blocks. The greatest virtue of Scratch is to allow to create stories, games or animations without previous programming

[1] https://scratch.mit.edu/

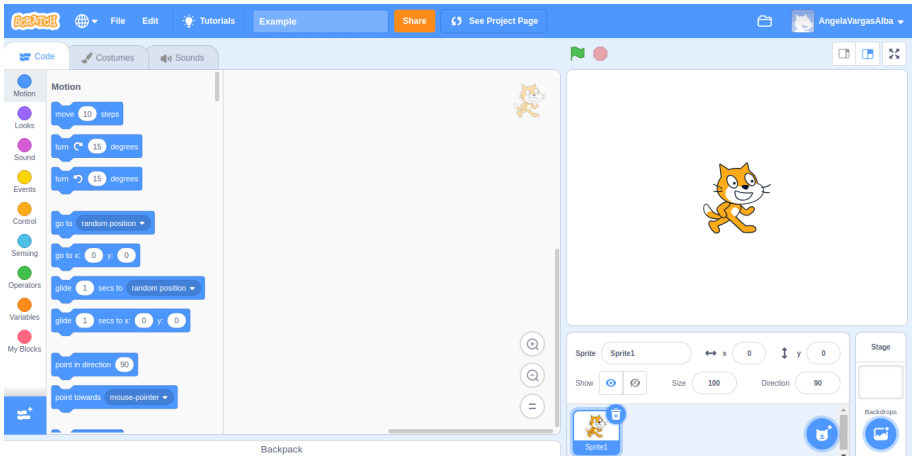


Figure (1.1) Editor design of Scratch 3.0.

knowledge.

Instead of using the conventional code, Scratch uses different pieces or blocks. Blocks are grouped by clearly identified categories, with different colors. The functionality is based on the creation of structures with these blocks and the generation of the instructions necessary for the program to work. In this way, learning to program has become an easier and more entertaining task.

After more than five years since the release of Scratch 2.0, the Kindergarten group at MIT launched Scratch 3.0 in January of 2019, a new version which included many new features. With this update, Scratch has improved different functionalities and characteristics, such as the variety of blocks, the web interface design, the sound editor, the extensions, the tutorials or the software, among others [Nin, 2019]. Its new editor design is shown in Figure 1.1.

On the other hand, Scratch is a world community. In this collaborative space, the "scratchers" can interact with each other or share and remix their projects. In addition, they can comment other projects or explain their doubts in a forum. In this way, beginner programmers can learn in a dynamic way, with the help of other programmers with more experience. In Figure 1.2 we can observe its statistics in October 2019. Scratch has become an amazing community which also keeps growing.

Figure (1.2) Statistics of the Scratch community in October 2019.

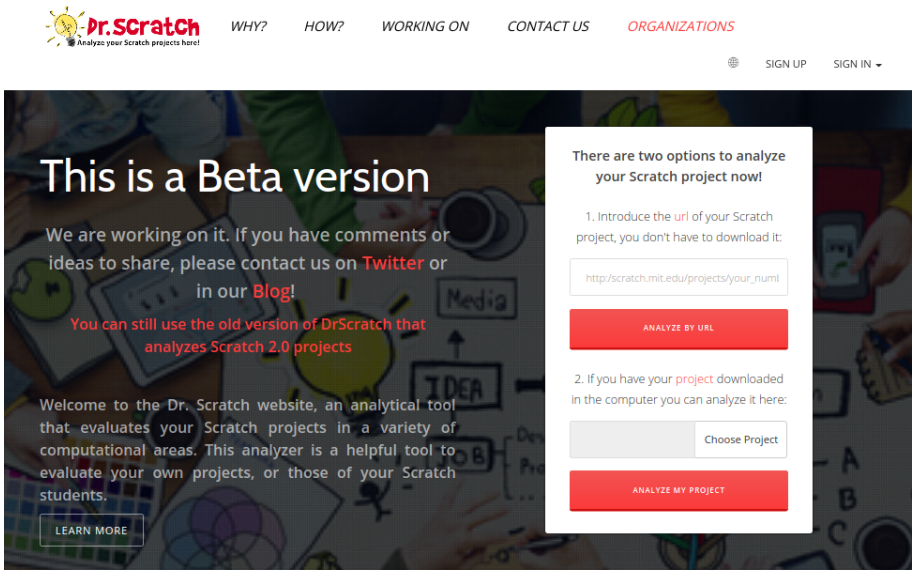


Figure (1.3) Web interface of the Dr. Scratch tool.

1.3 Frame of reference

With the growth of the Scratch community and its impact in the educational field, my tutor of this project -Dr. Gregorio Robles- together with the Dr. Jesús Moreno, observed the need of a tool to evaluate the Scratch projects. In this way, Dr. Scratch[2] was created in 2014.

Dr. Scratch is a web application which allows both teachers and students to automatize the analysis of Scratch projects. In this way, it helps to verify whether the projects have been programmed correctly, to analyze the bad practices in the code, to learn from their errors, to receive feedback or to improve the code. Its main web interface is shown in Figure 1.3.

For the analysis of the projects, Dr. Scratch used Hairball[3] in its initial version, a plugin

[2]http://www.drscratch.org/

[3]https://github.com/jemole/hairball

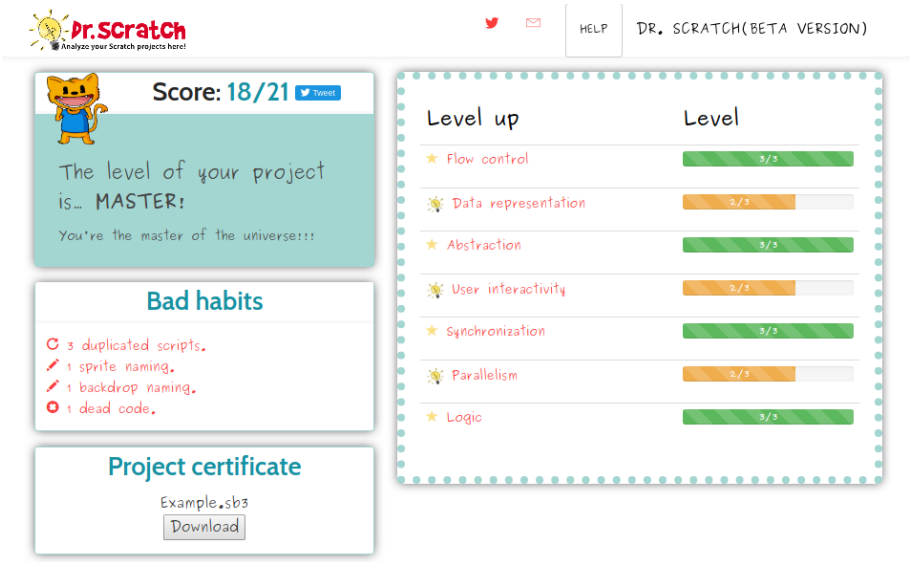


Figure (1.4) Dashboard of Dr. Scratch with the results of the analysis.

framework which analyzed the files generated with Scratch. These files have a compressed format and contain a JSON file with the relevant information about the blocks utilized in the Scratch projects. In the previous version, Scratch 2.0, the extension of these files was *.sb2*. This format was replaced by the extension *.sb3* in Scratch 3.0. In this way, the content of these JSON files would change significantly and Hairball would not be compatible.

Levering the need to modify the functionality of Hairball to adapt it to the new JSON format, we decided to remove it. The essential objective of this step was to simplify the software of Dr. Scratch, removing Hairball and integrating its new features in the own code of the tool.

Therefore, the starting point of this project was to adapt Dr. Scratch to the new version Scratch 3.0, which would be launched in the coming months. The challenge of this process was to get the same functionalities, but without the Hairball module.

Once the projects are analyzed, Dr. Scratch shows different dashboards with the results of the analysis, which can be observed in Figure 1.4.

In order to measure the development grade of CT demonstrated in programming, Dr. Scratch gives a numeric punctuation based on the level reached in each of the seven categories of CT: abstraction, logical thinking, synchronization, parallelism, flow control, user interactivity and data representation [Moreno-León et al., 2015]. Each of these capacities receives a punctuation from 0 to 3 points, depending on the Scratch blocks used. In this way, the final score varies

between 0 to 21 points, differentiating three different levels of programmers: Basic (0 to 7 points), Developing (8 to 15 points) and Proficiency (16 to 21 points).

In addition to the CT analysis, Dr.Scratch includes other key functionality in the development of this project: the analysis of bad smells that may exist in the Scratch project. That is, code that has never been executed (dead code), code that is copied and pasted in the scripts of different characters (repeated code) or the lack of significant naming (default naming). However, this dashboard was shown in a small space to the left and in a very summary way.

Another main challenge of this project was to change the importance that bad smells had in Dr. Scratch, in order to raise awareness of their importance. To carry out this process in the most appropriate way, we developed previously an exhaustive analysis about the behaviour of bad smells in Scratch projects.

With this analysis, we could verify that bad smells are present in the majority of the Scratch projects and that they had a negative impact in the development of CT skills. From this point, after a deeper and more exhaustive analysis of them, we considered necessary the development of a new specific model for bad smells in Dr. Scratch. Its final design and implementation can be observed in Figure 1.5.

Finally, we found an important weakness in Dr. Scratch during the analysis: It scores positively the presence of bad smells in the Scratch projects. In other words, a project with a larger number and variety of blocks -even these blocks are repeated or dead, for example- will have more punctuation than a simpler project -even it does not have any bad smell.

From this discovery, we wanted to prove the effectiveness of our analysis with other scenario: the assessment of bad smells in Scratch projects with teachers. In this way, we wanted to verify if teachers develop the same behaviour which we found in Dr. Scratch.

We developed an experiment in which two teachers of secondary school evaluated six Scratch projects. The experiment was composed of two phases. In both of them they had to analyze and evaluate the same Scratch projects, but firstly in a general way -with any information about bad smells- and, secondly, in a more guided way -receiving information about bad smells.

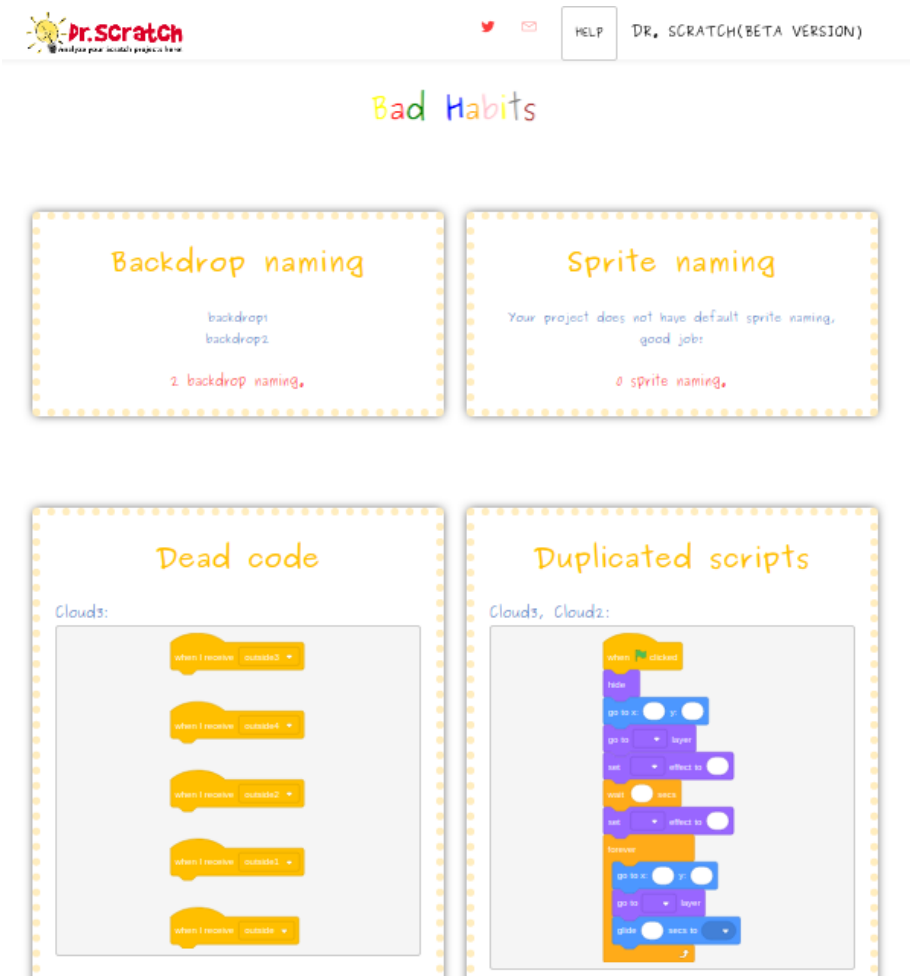


Figure (1.5) Interface design for the new model of bad smells in Dr. Scratch.

1.4 Motivation

Nowadays, there is a lack of resources to introduce CT, in particular programming, in the educational field. In many countries is already being implanted this area as a mandatory subject in schools. However, a lot of teachers do not have enough formation to teach in the most appropriate way this academic world.

Teaching children some good methods of programming and how to avoid bad practices, is one of the most difficult tasks to begin the development of CT. Nowadays, in an era in which programming is at its peak and there is a need to introduce it since childhood, it is very important to do in a responsible and appropriate way since the beginning.

For this reason, the main motivation of this work has been to analyze in detail the behaviour of bad smells and its evolution. A suitable study about bad smells can help to raise awareness about its presence both in students and teachers, and fostering a more appropriate learning guide.

1.5 Structure of the dissertation

In order to facilitate the reading of the dissertation, its structure is described in this section. This work is divided into six chapters, which are detailed as follows:

- Introduction. This chapter includes a description about the frame of reference of the project, as well as the context in which is involved. Moreover, the main motivation for its realization is explained.

- Objectives. In Chapter 2, the general objective is divided into four main different parts. The specific objectives to reach it are also detailed. In addition, a project timeline is included to clarify the planning and organization of all of them.

- State of the art. We describe the main tools and technologies used during the project in this chapter. In order to facilitate the understanding of them, we have grouped the tools according to the different phases of the project.

- Design and implementation. This chapter includes the progress of each phase of the project and the implementation of the different tools described. The development of the

bad smells analysis, as well as the previous and subsequent processes that have been needed, will be detailed in this section.

- Results. In this chapter, we mainly describe the results obtained from the bad smells analysis. We answer the research questions raised in the project and analyze the results found. In addition, we show the results from the assessment experiment developed.

- Conclusions. In this final chapter, we draw conclusions and hint to ideas that could be used for further research.

Chapter 2

Objectives

In this chapter, the considered objectives throughout this project are described. In addition, a timeline planning by means of Gantt Chart is included in order to detail the periods of time which have been dedicated to each of them.

2.1 General objective

The general objective of this project is composed of four clearly differentiated parts, which are described below:

- To update of the design and software of the web tool Dr. Scratch in order to adapt it to the new version of the programming language Scratch.

- To develop an analysis of the bad smells detected in the Scratch projects with the support of the Dr. Scratch tool.

- To implement a new model of visualization for these bad smells in order to raise awareness of their presence and importance.

- To develop an assessment experiment with different teacher profiles in order to evaluate the effectiveness and impact of our study.

2.2 Specific phases (and sub-objectives)

In order to describe the subobjectives, in the specific phases of the project, that were necessary to achieve each part of the general objective proposed, it is important to detail what was the starting point of the project.

At the beginning of the project, a version of Dr. Scratch already existed. This version analyzed Scratch 2.0 projects and returned different dashboards with the results. However, the work team of Scratch started to announce the launch of a new version of their language in the coming months. From this point, the specific objectives were divided into five different phases as follows:

1. Documentation and training

    - Documentation and training about the work context: Computational Thinking, Scratch, Dr. Scratch, bad smells in programming, Hairball or cloud tools, among others.
    - Presentation at the 3rd Scientix Conference: development of an initial paper about Dr. Scratch and its presentation at the congress developed in Brussels, Belgium.

2. Update of Dr. Scratch

    - To launch the existing version of Dr. Scratch: the version of Django was obsolete. It was necessary to update Django in order to start working with Dr. Scratch.
    - To update the code: to remove the Hairball and Kurt modules. For this objective, it was necessary the development of some scripts with the same functionality than the modules and their subsequent integration in the code of Dr. Scratch.
    - To add new languages: to include the Russian language as an option in the new version of Dr. Scratch.
    - To create a testing platform in production: to launch a first version of the Dr. Scratch update in a testing platform in Azure, during a month, and to correct possible mistakes during this period.
    - To coordinate both versions of Dr. Scratch: to have both versions of Dr. Scratch in the same platform of Azure, working in a compatible way in the official web of Dr. Scratch.

- To stabilize and maintain the new version: to correct errors and to improve some functionalities.
- To migrate the new version of Dr. Scratch to the Google Cloud Platform.

3. Analysis of bad smells

   - To collect data and create the files for a preliminary analysis of bad smells: in order to initiate the analysis, it was necessary to analyze a set of projects with Dr. Scratch, as well as a later process of treatment and filtering of the data.
   - To perform an analysis of bad smells: an exhaustive analysis of the data set of analyzed projects, mainly with Pandas software.
   - To present the first results in the TACKLE Congress: a presentation about the results found in the general analysis of bad smells.
   - To improve the analysis of bad smells: to collect new data for a second analysis and to improve the previous one.
   - To develop a more specific statistical analysis of bad smells: to implement a new treatment data for a deeper analysis with the t-student test.

4. Development and implementation of the bad smells model

   - To improve the functionality of bad smells in Dr. Scratch: To achieve the analysis of a bigger number of cases about bad smells, such as loop dead code or backdrop naming by default, among others.
   - To implement a new model based on bad smells in Dr. Scratch: to develop a new dashboard in which bad smells are shown in a more visual and clearer way.
   - To integrate the model in the code of the Dr. Scratch tool.

5. Development of an assessment experiment

   - To develop an assessment experiment about the bad smells evaluation with different teacher profiles.
   - To analyze the results of the experiment: to compare the results obtained with those of the previous analysis of bad smells, in order to check the effectiveness and impact of the study.

2.3 Project timeline

This project has been carried out during two full years. In order to organize the proposed objectives over time, we identify objectives in the short, medium and long term. In this way, the work is divided into the five phases described above:

- Phase 0: Documentation about the context, in order to understand the code and fix some bugs.

Figure (2.1) Timeline planning for phase 0.

- Phase 1: Update of the Dr. Scratch tool.

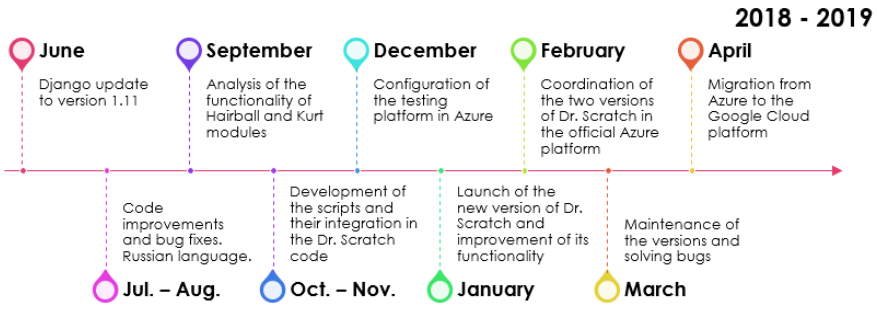


Figure (2.2) Timeline planning for phase 1.

- Phase 2: Analysis of bad smells
- Phase 3: Development of the bad smells model
- Phase 4: Assessment experiment

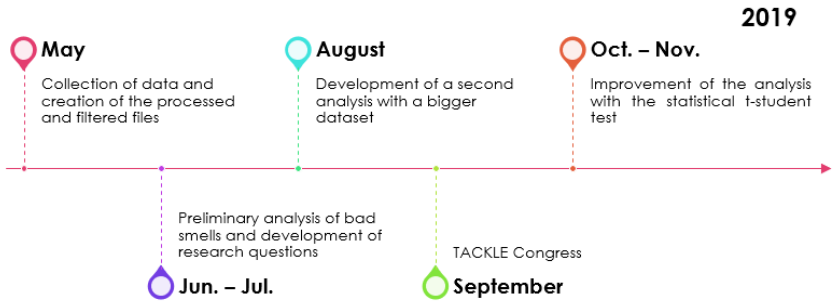


Figure (2.3) Timeline planning for phase 2.

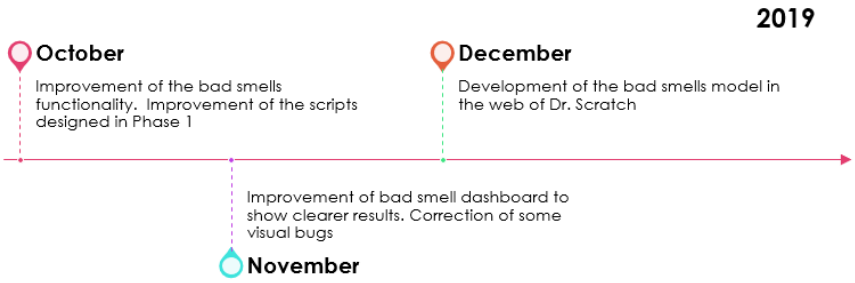


Figure (2.4) Timeline planning for phase 3.

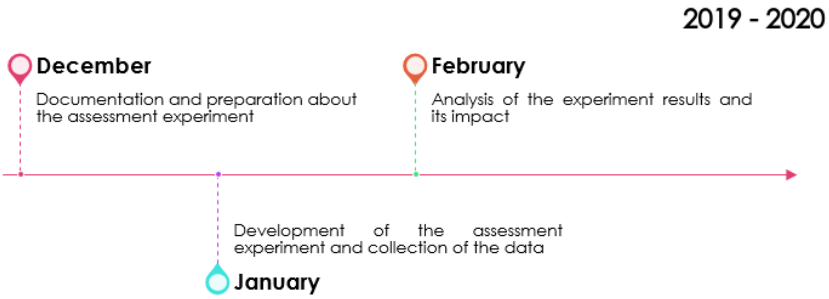


Figure (2.5) Timeline planning for phase 4.

Chapter 3

State of the art

In this chapter, the main technologies and tools used in the development of the project are described. Due to this work is composed of several different phases, the technologies utilized belong to different fields. For this reason, the definition of each of them will be categorized in each of the described phases of the project. Figure 3.1 presents the distribution of the phases throughout the project and the technologies used in each of them.

3.1 Phase 1. Update of the Dr. Scratch tool

3.1.1 Python

Python[1] is a high-level general purpose computer programming language, suitable for web application implementation [Kuhlman, 2009]. It is an open source language, optimized for software quality, developer productivity, program portability and component integration. Thanks to its characteristics, Python is used by hundreds of thousands of developers around the world and considered to be among the top four of five most widely-used programming languages in the world [Lutz, 2010]. The code of Dr. Scratch is programmed with the version 2.7 of the Python language. Some of its main features are as follows [JavaTpoint, 2018]:

- Easy to learn and use. Python is developer-friendly and high level programming language.
- Expressive language. Python is understandable and readable.

[1] https://www.python.org/

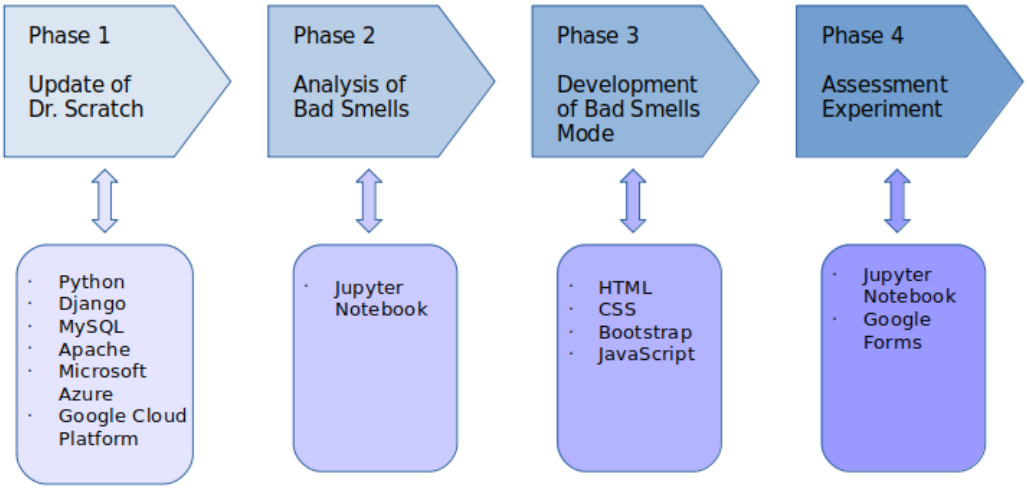


Figure (3.1) Distribution of the technologies for each phase of the project.

- Interpreted language. Python executes the code line by line at a time. This makes debugging easy and thus suitable for beginners.
- Cross-platform language. Python can run equally on different platforms such as Windows, Linux, Unix and Macintosh, among others. So, Python can be considered as a portable language.
- Object-Oriented language. Python supports object oriented language.
- Extensible. It implies that other languages such as C/C++ can be used to compile the code and, thus it can be used further in our python code.
- Integrated. It can be easily integrated with languages like C, C++, JAVA, among others.
- Large standard library.

3.1.2 Django

Django[2] is a free and open source web application framework, written in Python. A web framework is a set of components which helps to develop websites faster and easier [DjangoGirls, 2020]. Django was designed to support developers to implement applications from the concept to com-

[2]https://docs.djangoproject.com/

pletion, as quickly as possible. There is an excellent documentation with several tutorials, guides or models about Django.

Django is based on the MVC pattern (Model-View-Controller), a software architecture pattern which separates data presentation from the logic of handling user interaction. The *model* handles data representation and is defined in *models.py*. It serves as an interface to the data stored. The *view* part represents the answer generated by a controller, commonly the HTML of a web page. Regarding *controller*, it provides the logic and is composed of two scripts: *urls.py*, which manages the petitions to different views, and *views.py*, which generates the HTTP response.

In this work, the versions implemented in the Dr. Scratch tool were 1.7 and 1.11 (Long-Term Support Release).

3.1.3 MySQL

MySQL[3] is a Relational Database Management System (RDBMS) based on Structured Query Language (SQL). MySQL is based on a client-server model and the server, which is the core of MySQL, handles all of the instructions. The Dr. Scratch tool works with this database because it is associated with web applications and online tools. The main characteristics of MySQL are as follows:

- MySQL is an open source software.
- High performance and scalability to meet the demands of growing data loads and users.
- Multi platform. Platform independence, giving the flexibility to develop and deploy it in multiple operating systems.
- Flexible and easy to use.
- High standard security.

[3]https://www.mysql.com/

3.1.4 Microsoft Azure

Microsoft Azure[4] is a set of cloud services which allow to build, manage, and deploy applications on a global network using different tools and frameworks. Thanks to its online platform, it is easy to develop different services, such as computing, storage, analytical or networks. Dr.Scratch tool was hosted on a Virtual Machine in the Azure platform, but the new version was migrated to another cloud service due to different reasons which will be detailed in Chapter 4. The Virtual Machines of Azure allow the implementation of tools and applications in a virtual system. Some of the reasons why Azure was chosen were:

- Scalability. Possibility to personalize the required services in a Virtual Machine (type of machine, capacity, performance, among others).
- Flexibility and payment for use. Easy deployment.
- Compatible with MySQL technology.
- High security.

3.1.5 Apache

The Apache HTTP Server is a free and open source software web server. A server is necessary to show the content of the tool as a web page. Apache is a multi platform software, it works both in Unix and Windows servers. The server and the client communicate trough the HTTP protocol and Apache is the responsible for guaranteeing a fluency and safety communication between both extremes.

Apache is based on different modules and files that we had to configure to deploy the Dr. Scratch tool on the web. Thanks to its structure, it is highly personalized and we could host both versions in the same domain. One of the most important modules is the *mod_wsgi* module, which allows to connect Django and Azure. The current version that is working in Dr. Scratch is *Apache/2.4.29*.

[4]https://portal.azure.com/

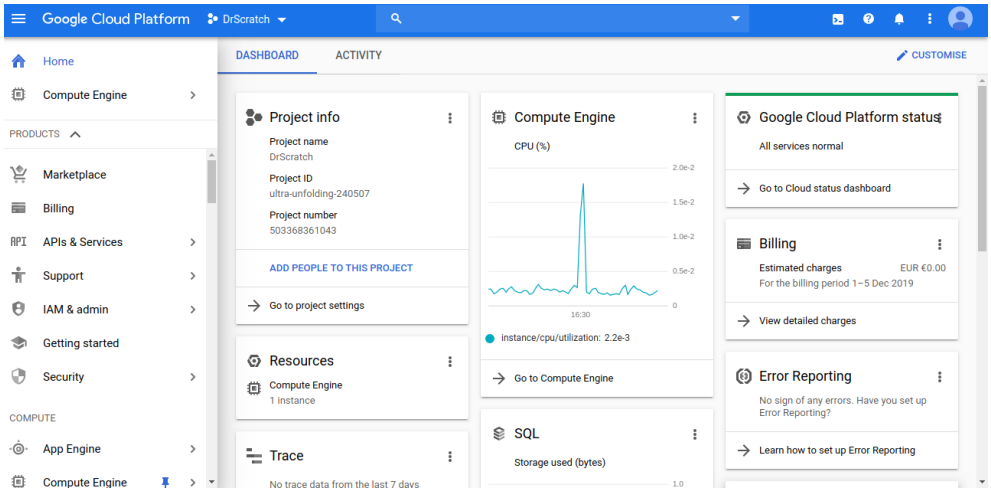


Figure (3.2) Dashboard of the Google Cloud Platform for the Dr. Scratch tool.

3.1.6 Google Cloud Platform

Google Cloud Platform (GCP)[5] is a set of physical resources, such as computers and hard disks, and virtual resources, such as Virtual Machines, which are in the data centers of Google around the world. The location of these data centers allows the distribution of the resources and consequently, a higher performance (because the services are closer to the clients).

There are a lot of available services in GCP, and the list is still growing. Thanks to its console, it is easy to manage and develop the necessary services for the Dr. Scratch tool. For this reason, among others, the new version was migrated from Azure to a new Virtual Machine of GCP. In Figure 3.2, we can observe its main console.

3.2 Phase 2. Analysis of bad smells

3.2.1 Jupyter Notebook

Jupyter Notebook[6] is an interactive working environment which allows, in a dynamic way, the integration of Python code, text, graphics and pictures in the same document. It is useful for statistical analysis, machine learning or data management, among others.

Some of the libraries which Jupyter includes and we have implemented in the analysis are as follows:

[5]https://cloud.google.com/
[6]https://jupyter.org/

- *Pandas*[7] is an open source library which provides high-performance, easy-to-use data structures and data analysis for the Python language. Its primary data structure is *DataFrame*, a potentially heterogeneous data structure with labeled axes (rows and columns). It has been essential for the management of the analyzed data set.

- *NumPy* is the fundamental package for scientific computing. It contains a powerful N-dimensional array object, sophisticated functions or useful linear algebra, among others.

- The *matplotlib.pyplot* library. It allows multiple options to work with graphics or figures. It has been necessary to represent the data and results.

- The *scipy.stats* module contains a large number of probability distributions, as well as statistical functions. In particular, we have implemented in this study the t-student test for the statistical analysis.

3.3 Phase 3. Development of bad smells model

3.3.1 HTML

HTML (HyperText Markup Language) is a language designed for web tools. The first version of HTML was launched in 1991, by the British scientist Timohty John Berners-Lee and it contained few items.

HTML is a language composed of different elements, tags and attributes. Thanks to its tree structure, it is possible to describe the content of a page in plane text. In the following example we can observe how the content is composed of opening tags, possible attributes and closing tags.

```
<!DOCTYPE html>
<html>
    <head>
        <title> Dr. Scratch </title>
    </head>
    <body>
        <div id="myExample">
```

[7] https://pandas.pydata.org/

```
            <p> This is a paragraph </p>
        <div>
    </body>
</html>
```

The first official standard HTML 2.0 was released in 1995. Actually, we have used the last version published in 2014, HTML 5. All the interface of Dr. Scratch is designed with templates of HTML.

3.3.2 CSS

CSS (Cascading Style Sheets) is a language used to define design styles for web pages. Mainly, it is used to describe the style of HTML files, such as the design, colors or fonts. CSS has a simple syntax and it can control multiple pages all at once. The same way, CSS is composed of elements, tags and attributes. We can observe an example of its structure.

```
#example {
    height: 100%;
    margin: 0 -60px;
}

body{
    font-family: sans-serif;
    background-color: red;
}
```

3.3.3 Bootstrap

Bootstrap[8] is an open source framework for developing with HTML, CSS, and JavaScript. It facilitates the design and appearance of the web tools by means of CSS libraries which include buttons, typography or menus, among others. Bootstrap is responsive, that is, it allows to adapt the web pages to any type of devices and screens. In addition, the Bootstrap platform offers a good documentation and a large number of free templates with the implementation of the design of a complete web page.

[8]https://getbootstrap.com/

3.3.4 JavaScript

JavaScript is a programming language which works on the client side. It is an interpreted language, which means it can be incorporated in a web page without the need of compilation. JavaScript code runs directly in a web browser and it controls the dynamic elements of the web page.

Although the JavaScript code can be hosted in external files, usually it is located between the tags of the HTML document:

```
<body>
    <script>
        JavaScript Code
    </script>
</body>
```

3.4 Phase 4. Assessment experiment

3.4.1 Google Forms

Google Forms[9] is a tool which allows the creation of simple forms or questionnaires. It collects data from online surveys which can be obtained in data sheets. It is a way to gather information quickly and easily.

Thanks to this tool, it was possible to develop, in a simple way, some questionnaires for the different teacher profiles who implemented the experiment. We obtained different answers and data from Google Forms and then we could analyze them easily.

[9] https://www.google.es/intl/es/forms/about/

Chapter 4

Design and implementation

In this chapter, all the phases of the project mentioned above are detailed. The evolution of each process and its description, as well as the problems encountered, are described in each section.

4.1 Documentation and training

The beginning of the project was mainly based on a period of training and documentation. During the first months of this work, as we described in Chapter 2, it was necessary a previous formation in aspects such as CT, the Scratch language, Dr. Scratch and its architecture or the concept of bad smells, among others.

After this process, we decided to present an article about the Dr. Scratch tool in the 3rd Scientix Conference[1] celebrated in Brussels, Belgium. This paper, "Dr. Scratch: Assess Scratch projects for Computational Thinking skills", was composed of a single page and summarized the general definition of Dr. Scratch and its main characteristics. This congress entailed the first contact with this researching project.

4.2 General architecture of Dr. Scratch

Dr. Scratch is a client-server tool which analyzes the JSON file of a Scratch project. Its main architecture can be shown in Figure 4.1. Clients make HTTP requests to the server, which was allocated in a virtual machine of the Azure platform with Ubuntu 14.04.2 LTS as operative

[1] http://www.scientix.eu/es/conference/scx3

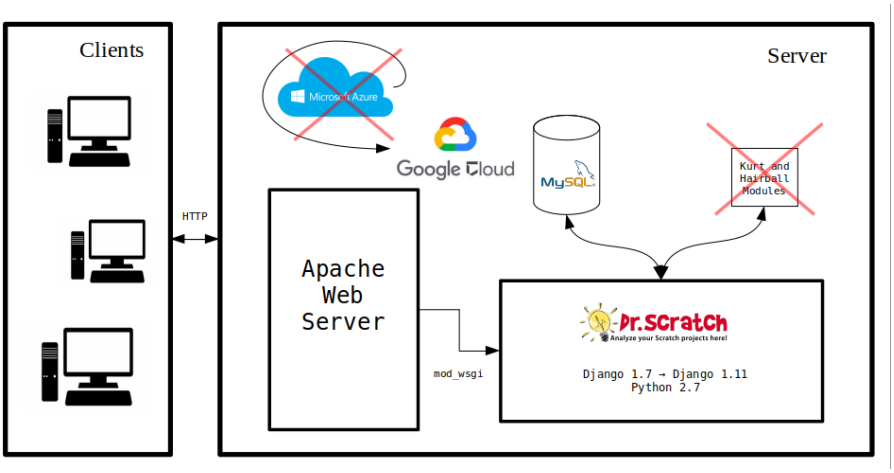


Figure (4.1) General architecture of Dr. Scratch.

system. It was possible thanks to a free subscription of Microsoft. However, this subscription was over when we were developing the new version and it had to be migrated to the Google Cloud Platform, which will be detailed in Section 4.2.3.

The version 2.4.10 of Apache Web Server was installed inside of the virtual machine of Azure, the same as the *mod_wsgi* module, which connects Azure with Django.

On the other hand, in order to collect the information, Dr. Scratch used the version 5.5.43 of the MySQL database.

Hairball and Kurt modules were also installed in the virtual machine because they were necessary for the analysis of the Scratch projects. Kurt is a Python library useful for working with the Scratch project files. Hairball analyzes the JSON file of the Scratch projects and returns the assessment of the CT development and the total number of bad smells that the project has. Hairball analyzes the blocks of the projects through different plugins.

4.2.1 Dr. Scratch 3.0

The starting point of this phase was the update of the Django version. The version 1.7 was obsolete, so we migrated the code to Django 1.11 LTS. This process was mainly based on minor modifications to the code and it could be done easily and quickly.

On the other hand, the Scratch community announced the launch of the new Scratch 3.0 version in the coming months. From this point, we investigated its main changes.

The most important modification was the structure of the JSON file of the Scratch projects. Hairball and Kurt modules were not compatible with the new format. Therefore, the first step

was to modify these modules, since they only supported Scratch 1.4 and Scratch 2.0 versions.

In order to simplify the software of Dr. Scratch, we decided to remove the modules instead of updating them to the new version -because we thought that it was an easier and more efficient solution. In its place, we programmed some new scripts with the same functionality, but integrated into the main code. In other words, the objective of this process was to develop different scripts which analyzed the CT development and the total number of bad smells in the Scratch projects and integrate them into the Dr. Scratch code.

Once that the functionality of Hairball and Kurt was replicated, we integrated the scripts in the main code of Dr. Scratch and removed the modules. We describe in more detail each of the scripts below.

Mastery

The *analyzer.py* script analyzes the blocks of the JSON file and returns the punctuation of each of the seven categories of the CT which the Hairball module analyzed, the total mastery and the level of competence of the project. Each category -flow control, data representation, abstraction, user interactivity, synchronization, parallelism and logic- is scored from 0 to 3 points, depending on the type of blocks. The total punctuation is the sum of the points of each category. Therefore, the range of the total mastery is from 0 to 21 points. Depending on the final mastery, the script differentiates three types of profiles (the same as Hairball module): Basic (0 to 7 points), Developing (8 to 15 points) and Proficiency (16 to 21 points). In Table 4.1 is detailed the evaluation of the competence levels for each CT concept [Moreno-León et al., 2015].

Sprite naming

The *spriteNaming.py* script analyzes the presence of names by default in the sprites (Sprite1, Sprite2, ..., SpriteN) in the JSON file of the Scratch projects. It returns the total number of default sprite names found in the program, as well as a list with all of them.

Backdrop naming

The *backdropNaming.py* script analyzes the presence of default names of the backdrops (Backdrop1, Backdrop2, ... BackdropN) in the JSON file of the Scratch projects. It returns the total number of this bad smell and a list with the default names of the backdrops found.

CT Concept	Competence Level		
	Basic (1 Point)	**Developing** (2 Points)	**Proficiency** (3 Points)
Flow control	Sequence of blocks.	Use of *repeat* or *forever* blocks.	Use of *repeat until* block.
Data representation	Modifiers of sprites properties (looks, motion).	Operations on variables.	Operations on lists.
Abstraction	More than one script or more than one sprite.	Definition of blocks.	Use of clones.
User interactivity	Use of *green flag* block.	Use of *key pressed*, *sprite clicked*, *ask and wait* or *mouse* blocks.	Use of video or audio sensing blocks.
Synchronization	Use of *wait* block.	Use of *broadcast* and *receive messages*, *stop all* or *stop program* blocks.	Use of *wait until*, *when backdrop change to* or *broadcast and wait* blocks.
Parallelism	Two scripts on *green flag*.	On the same sprite, two scripts on *key pressed* or on *sprite clicked*.	Two scripts start with the same multimedia event, *when broadcast received*, *when backdrop switches to* or *create clone*.
Logic	Use of *if* block.	Use of *if else* block.	Use of operator blocks.

Table (4.1) Competence levels for each CT concept.

Duplicated code

The *duplicatedScripts.py* script analyzes the repeated block structures. It detects scripts, with more than five blocks, which are repeated in the same sprite or in a different one. It returns a dictionary in which the keys are the name of the sprites that have repeated code and the values are the duplicated scripts, respectively. In addition, it returns the total number of duplicated scripts found.

An example of the response obtained is shown below. In this case, there are three sprites -*Elephant*, *Bear* and *Penguin*- which contain in their code the same script duplicated, composed of five blocks.

```
1 duplicated script found
{'Elephant, Bear, Penguin': [['event_whenbroadcastreceived',
'looks_say', 'looks_switchcostumeto', 'looks_show', 'looks_hide']]}
```

Dead code

The *deadCode.py* script analyzes unreachable code in isolated blocks or in block structures in the Scratch project. The different situations that it may detect are:

- Empty loop blocks: control blocks -*forever*, *wait* or *if else* blocks- which do not have other blocks inside.
- Empty conditions: control blocks which do not have conditions to execute the code inside of them.
- Hat blocks: structures which do not start with an event block, such as *green flag* or *key pressed*, among others.
- Not message: scripts which start with the *when broadcast received* block, but this message is never sent.
- Dead code: blocks that do not have parent or next blocks. Isolated blocks that are never executed.

Finally, this script returns a list of the dead blocks (scripts or isolated blocks) for each sprite and the total number of this bad smell.

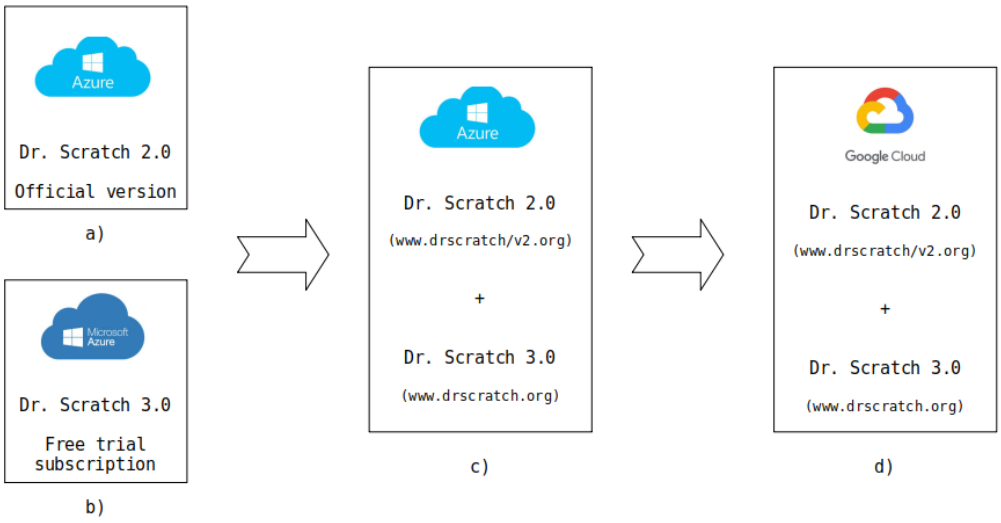


Figure (4.2) Migration process for the new version of Dr. Scratch.

Attribute initialization

The *attributeInitialization.py* script tried to replace the Hairball plugin that checked if modified attributes were properly initialized. There are many blocks related to attributes in Scratch: costume, orientation, position, size and visibility. Due to the complexity of the code to analyze whether all of these properties were correctly initialized, it was so complicated to develop the same code in a script. Finally it did not work properly and the new script did not replace perfectly the plugin. For this reason, in the new version of Dr. Scratch this bad smell is not included.

Lastly, in addition to the Django update and the integration of the scripts in the Dr. Scratch code, we developed other functionality: we included a new language in the new version of Dr. Scratch, the Russian. It was a relatively simple process thanks to the collaboration of a Russian translator who worked for the Scratch community. Once that we received the file with the Russian translation, we included it in the Dr. Scratch tool with small changes in its code.

4.2.2 Migration of Dr. Scratch 3.0 to Azure Platform

Before the official launch of the new version of Dr. Scratch, we decided to test it during a period of time in another virtual machine of Azure. In Figure 4.2 we can observe the process from the development of the new version until its current deployment.

Thanks to an educational subscription, we had a free trial period of the services offered by Azure during a month. In this way, we configured a virtual machine with the new version of Dr. Scratch and we tested it during a month, as we can observe in Figure 4.2 b). To carry out this process, it was necessary to configure Apache throughout the file *httpd.conf*. This file contained the instructions for the main settings. In addition, we needed a testing database with MySQL.

After correcting some small errors during the trial period, such as the download of certificates or the change of language, we configured the official virtual machine of Azure with both versions of Dr. Scratch, as we observe in Figure 4.2 c). Again, it was possible thanks to the configuration of the *httpd.conf* file of Apache.

Finally, in January 2019 we launched the new version. Dr. Scratch 3.0 was allocated in the *www.drscratch.org* domain. Its main page contained a link which redirected to the old version, allocated in the *www.drscratch.org/v2* domain. We can observe the main interface of the tool in Figure 1.3.

4.2.3 Migration of Dr. Scratch 3.0 to Google Cloud Platform

In April 2019, the funding which we received from Microsoft was over and we studied different options to allocate Dr. Scratch. Finally, the best solution found was to migrate the application to the Google Cloud technology, as we can observe in Figure 4.2 d).

We received a grant from the GCP Research and, thanks to that, we configured the new services. We deployed a new virtual machine with 18.04.4 Linux as operative system. In addition, we updated MySQL to the 5.7.28 version. We migrated both versions to the new platform and we configured again the Apache module. It was a quick process and we had a short trial period. After solving small problems, we got a stable version which is working at present.

4.3 Analysis of bad smells

Despite the new version Dr. Scratch 3.0 was stable in the Google Cloud Platform, all the analysis of bad smells was carried out with Dr. Scratch 2.0. The reason was that the data set used in the analysis was composed of projects designed with the old version of Scratch, that is, with the *.sb2* extension and its corresponding format.

Bad Smell Type	Definition	Impact on Learning
Duplicated scripts	Code is copy and pasted, sometimes with minor changes	It hinders the use of user-defined blocks and as such can be seen a limitation to the development of the abstraction skill
Default names	Objects are not given a meaningful name, but keep the default *SpriteN* name	It hinders interaction among objects, as using them in other objects becomes more difficult
Dead code	Code that is never being executed (usually because they do not have a starting condition)	It may indicate missing functionality
Attribute initialization	Variables are not well initialized	It hinders the start of some objects, because their position, size or costume, among others, are not correctly initialized

Table (4.2) Description of bad smells and their impact on learning.

Therefore, both data sets were analyzed with the Hairball module of Dr. Scratch 2.0. The Dr. Scratch tool in the previous version identified four different types of bad smells that can be present in Scratch projects [Robles et al., 2017]: copy and pasted code (duplicate scripts), the use of default names for sprites (default names), code that is never being executed (dead code), and variables that are not correctly initialized (attribute initialization).

Their description and characteristics, as well as their impact in the CT development, are summarized in Table 4.2.

	Number of projects	Number of snapshots
Data set a	771	62,074
Data set b	250,163	-

Table (4.3) Distribution of the data set used in the general preliminary analysis.

4.3.1 Data set description

In order to analyze the presence of bad smells, as well as their relationship with the level that users have in CT development, a large set of projects was necessary. In this work we have analyzed two different data set. The first of them, which we will call data set a, was analyzed in a preliminary and general analysis. In order to improve and verify the results obtained, we repeated the analysis with a bigger data set, which we will call data set b. The distribution of both data set is summarized in Table 4.3.

Data set a

Data set a was created and studied in another, previous research [Troiano et al., 2019]. A group of 438 students designed games for STEM using Scratch 2.0. During this process, the authors obtained snapshots of the process each minute during a period of time -only if the project registered changes-, in order to show a timeline evolution. The total number of projects without taking into account the replicas over time, was 771. The total number of snapshots over time were 62,074, with an average of 78 snapshots per project.

As a result, the complete data set was comprised of 62,074 samples formed by the different snapshots of each project[2]. The objective of storing snapshots was to analyze the same 771 projects in different points of time.

Data set b

Data set b was also created and analyzed in another previous research [Aivaloglou et al., 2017]. This data set contained 250,163 Scratch projects, from more than 100,000 different users of its community. These projects were scraped from the Scratch repository. The scraper and all the

[2]https://drive.google.com/drive/u/0/folders/1tDI6nx2f6344xJAKeUeWBeTg0YzxE3bO

project files are available in the GitHub repository *TU Delft Scratch Research Team*[3].

From this repository, we stored the useful data in two CSV files, *metadata.csv* and *code.csv*. These files included, for each Scratch project, its metadata -user name, total views or the project URL, among others- and information about its code -sprite types, block types, total blocks or CT development of the Dr. Scratch assessment, among others-, respectively.

4.3.2 Data collection

From this point, we needed to create our own CSV files from the data set described above. The process of the data set construction for the analysis is detailed below.

Data set a

In order to analyze the 62,074 snapshots with the Dr. Scratch tool, it was necessary to design a script. This script, *projects_analyzer.py*, was programmed in Python. Its main function was to open each folder with the Scratch project and analyze its JSON file. As we mentioned previously, the analysis was carried out with the Hairball module and all its plugins. The result obtained from the script was a CSV file, *evolution_time_results.csv*, with the information necessary for the analysis: the total mastery, the score of each category of the CT, data about bad smells and data about the blocks.

During the analysis process, we found 2,158 erroneous snapshots due to different reasons: the project was saved incorrectly, the code contained special characters or some field was empty, among others. In this way, the final set of valid samples for the analysis was comprised of 59,916 snapshots and 754 projects.

On the other hand, we programmed another script to analyze the latest registered version of each project. In other words, we designed a script which analyzed the 771 projects without the timeline evolution. The result obtained was another CSV file, *final_projects_results.csv*, with the same structure than *evolution_time_results.csv*, but storing only the last snapshot of each project. Again, we found the same erroneous samples and the final valid data set was comprised of 754 projects.

[3]https://github.com/TUDelftScratchLab/ScratchDataset

	Total samples	Valid samples	Wrong samples	Survival ratio
Projects	771	754	17	97.80%
Snapshots	62,074	59,916	2,158	96.52%

Table (4.4) Distribution of the final data set a used in the general analysis.

	Total samples	Valid samples	Wrong samples	Survival ratio
Projects	250,163	231,024	19,139	92.35%

Table (4.5) Distribution of the final data set b used in the general analysis.

The statistics and distribution of valid projects and snapshots for the analysis are summarized in Table 4.4.

Data set b

The size of the data set b was much bigger than the data set a. For this reason, we found complications when we tried to open the file *code.csv* and analyze it. Its analysis with a Python script was too slow and required too much storage space.

Finally, the solution was to create a Jupyter notebook and read and analyze the file line by line. In this way, we could join and select the useful information from the *code.csv* and *metadata.csv* files. The result of the Jupyter notebook was a new CSV file, *final_dataset.csv*, composed of the useful data from both files and with the structure which we wanted for the analysis: total mastery, the score of each category of CT and information about bad smells and the blocks.

During the analysis process, we found wrong projects again. We found null data in the *metadata.csv* file and special characters in the *code.csv*. Finally, the number of valid projects was 231,024. The statistics of the valid samples for the data set b are summarized in Table 4.5.

4.3.3 General analysis

The main objective of this phase was to analyze the presence of several bad smells in Scratch projects and how they relate to the development of CT skills. We carried out a preliminary

analysis with the data set a in order to analyze to what extent bad smells are present in Scratch projects and what is its impact. Then, we repeated the same analysis with data set b in order to verify the results with a bigger set of projects. We developed both analysis with Jupyter notebook, based on the CSV files generated in the previous section.

The starting point of the general analysis was to raise the main research questions that we wanted to analyze. All the proposal, as well as the development of the analysis and the results obtained, were published in the paper "Bad Smells in Scratch Projects: A Preliminary Analysis" [Vargas-Alba et al., 2019]. We developed this paper for the EC-TEL congress[4] and we presented it in the TACKLE workshop[5] (the 2nd Systems of Assessments for Computational Thinking Learning) in Delft, Netherlands.

RQ1. To what extent are bad habits present in Scratch projects?

In particular, we answer this question by offering the percentage of projects that have at least one type of bad smell. This question allows to see how frequent projects show a bad smell, hinting to the relevance of the topic. We expect a significant number of projects to contain bad smells.

RQ2. Does the development of CT skills relate to a minor presence of bad smells?

We would like to find out if the presence of bad smells correlates with the complexity of the projects. Our hypothesis is that projects that have higher degrees of CT development will have less bad smells, as these may hinder the development of CT skills.

RQ3. Do projects with more blocks have a higher number of bad smells?

More complex projects usually have more blocks. Thus having a single bad smell in a small, simple project may have less impact than in a project with hundreds of blocks. In the former case, the impact could be big, while in the latter it could be seen as an exception, with little impact.

To answer this question, for projects of the same level of CT development we calculate the ratio of the number of bad smells detected to the total number of blocks. We expect that this

[4]http://www.ec-tel.eu/

[5]https://sites.google.com/site/2019tackleworkshop/home

ratio decreases with an increase in the development of CT skills required to create the Scratch projects.

RQ4. Can we find a relation among specific bad smells?

As by now, we have considered all type of bad smells together. In this question, we dig into each of them separately. It may be possible that some bad smells appear more frequently in projects of lower complexity, while others appear in more complex projects.

RQ5. To which extent can bad smells be identified in each of the CT development phases?

Related to the previous question, we analyze how the different bad smell types appear in projects in the different stages of CT development. Therefore, we consider projects with a low complexity (basic), medium complexity (developing) and major complexity (proficiency) and compute how often they contain a specific type of bad smell.

We expect that several types of bad smells appear in the early phases (basic), while others are more prominent in more complex projects (proficiency). We assume therefore that learners that achieve higher levels of complexity have overcome certain bad smells due to having developed certain CT skills, while other bad smells appear in those more complex projects.

4.3.4 Statistical analysis

After the general preliminary analysis, we decided to deepen the study and develop a more specific analysis. In order to analyze in more detail the impact of bad smells in our variable of interest (CT development), we carried out a statistical analysis. The main objectives of this analysis were:

- To analyze the behaviour and distribution of each type of bad smell through Dr. Scratch, during the timeline of the projects.
- To analyze the impact on the development of the CT skills.
- To carry out a statistical *t-student* study in order to measure the effect that bad smells have throughout a period of time.

	Total samples	**Valid samples**	**Survival ratio**
Projects	754	543	72.02%
Snapshots	59,915	59,332	99.03%

Table (4.6) Distribution of the processed data set for the statistical analysis with deciles.

In order to develop these objectives, we compared two main groups or populations. The first of them was a set of projects with a few amount of bad smells and the other one, a set of projects with a huge amount of bad smells, in a given initial state. Afterwards, we wanted to know if they showed a different timeline evolution.

From this point, there were two hypotheses, *H0* -both populations presented no differences in the mean value of bad smells- and *H1* -the populations had different mean value of bad smells. We wanted to prove how random could be *H1*.

Sample processing

Therefore, this second study was based on data set a, since it was necessary a timeline evolution of the projects. To perform an effective analysis, we divided the total number of snapshots into deciles, that is, into 10 proportional periods of time (D0-D9), for each project. Afterwards, only those projects with at least 10 snapshots -at least one snapshot per decile- were considered as valid samples. In addition, we also removed those projects which had 0 points in the assessment with Dr. Scratch in D9. These projects did not have any development, so they were not useful for the study.

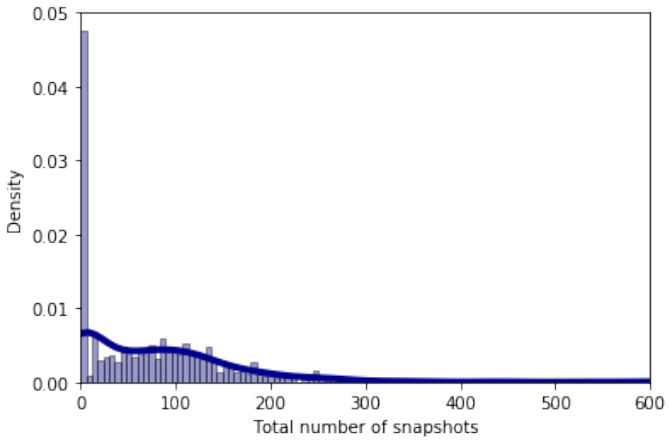
The final data set after removing invalid and null samples was composed of 543 projects and 59,332 snapshots. The survival ratio about valid projects decreases considerably due to a high density of projects with less than 10 registered snapshots. We can observe this effect in Figure 4.3. Its distribution is also shown in Table 4.6.

The distribution of the snapshots, for both participating projects and valid projects, is described, in a more detailed way, in Table 4.7.

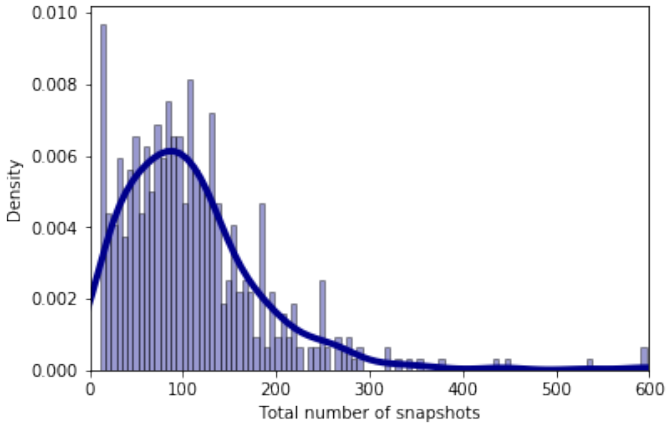
Finally, in order to avoid redundant information, we only stored the last registered snapshot of each project, in each decile. In this way, the data set for the analysis was composed of 543 projects and 5430 snapshots, with only one snapshot per decile.

	Minimal number of snapshots	Maximum number of snapshots	Mean of snapshots	Standard deviation
Participating projects	0	597	78.0	82.1
Valid projects	11	597	109.3	77.8

Table (4.7) Snapshot distribution in the processed data set for the statistical analysis with deciles.



(a) Participant projects



(b) Valid projects

Figure (4.3) Snapshot distribution for the statistical analysis with deciles.

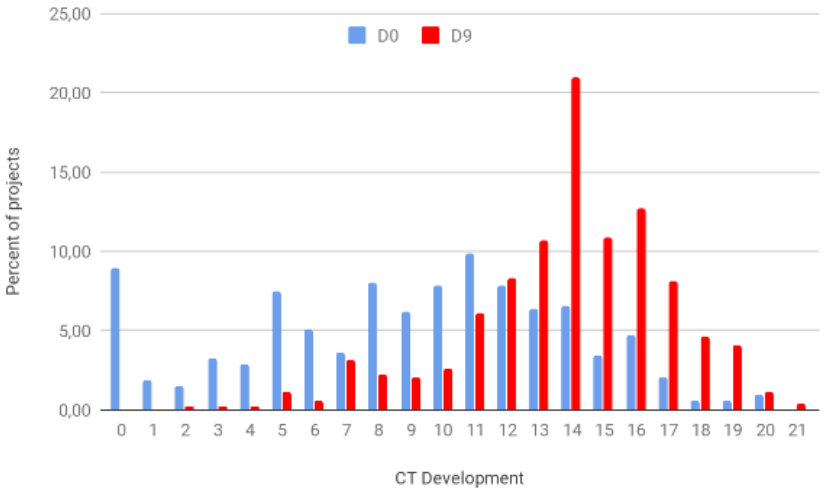


Figure (4.4) Distribution of the CT development for the valid data set in D0 and D9.

Characteristics of the sample

After dividing the valid data set into deciles, we analyzed its distribution according to the CT development obtained in the assessment with the Dr. Scratch tool. In Table 4.8 is summarized this information for D0 and D9, in order to show the evolution between the initial and final state of the projects.

We can observe how the density of the projects with a higher CT development is bigger in D9 than in D0. On the contrary, the density of projects with a lower CT development is bigger in D0 than in D9. This distribution implies a CT development throughout the timeline of the projects. We can observe this effect in a more visual way in Figure 4.4.

On the other hand, we also analyzed the distribution of the number of bad smells in each decile. In Figure 4.5 is shown the mean value of each bad smell for each decile. As we can observe, the number of bad smells increases with the timeline evolution. This result could be produced by the increase of the CT development of the projects and, consequently, with the increase of its number of blocks.

In order to verify this effect, we analyzed the ratio between the mean value of each bad smell and the total number of blocks in the projects, for each decile. In this way, we obtained the contrary effect, as we can observe in Figure 4.6. In this figure, the ratio decreases with each decile. In other words, the mean value of bad smells is lower when the CT development increases, as we expected.

Dr. Scratch CT	Frequency		Percent (%)	
Assessment	D0	D9	D0	D9
0	46	0	8.47	0.00
1	10	0	1.84	0.00
2	8	1	1.47	0.18
3	18	1	3.31	0.18
4	16	1	2.95	0.18
5	41	6	7.55	1.10
6	28	3	5.16	0.55
7	20	17	3.68	3.13
8	44	12	8.10	2.21
9	34	11	6.26	2.03
10	43	14	7.92	2.58
11	54	33	9.94	6.08
12	43	45	7.92	8.29
13	35	58	6.45	10.68
14	36	114	6.63	20.99
15	19	59	3.50	10.87
16	26	69	4.79	12.71
17	11	44	2.03	8.10
18	3	25	0.55	4.60
19	3	22	0.55	4.05
20	5	6	0.92	1.10
21	0	2	0.00	0.37
Total	**543**		**100**	

Table (4.8) Distribution of the CT development for the valid data set in D0 and D9.

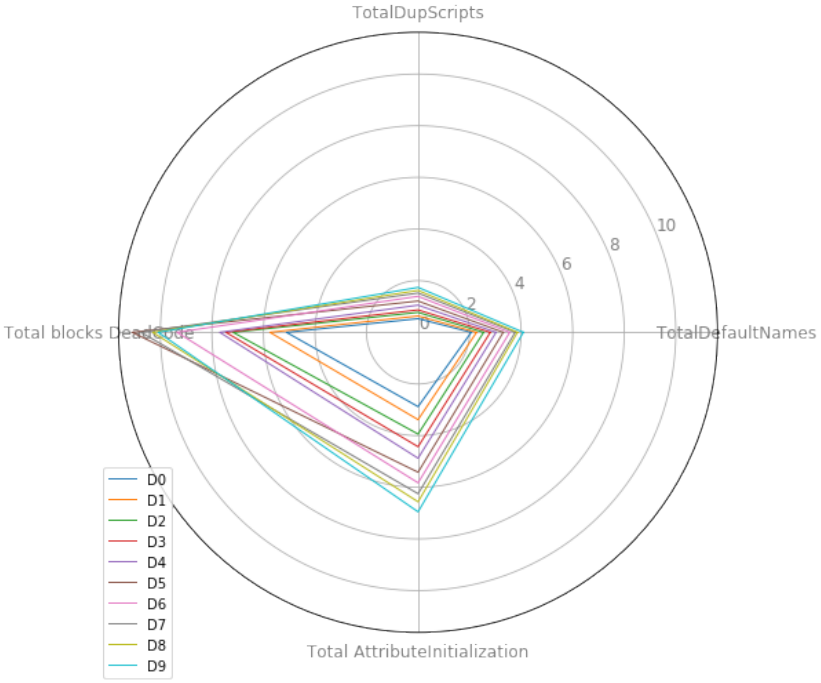


Figure (4.5) Distribution of the mean value of each bad smell for the deciles D0-D9.

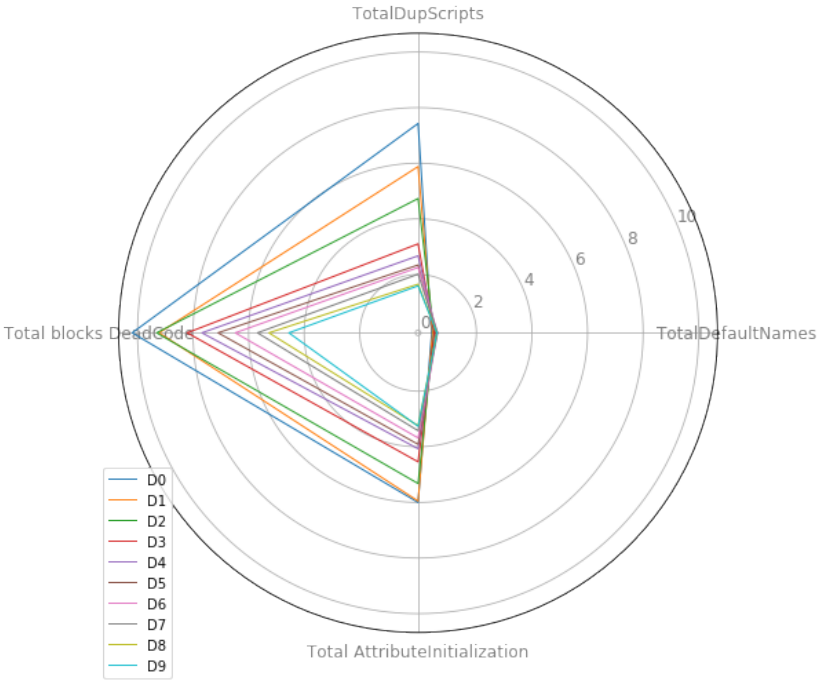


Figure (4.6) Mean value of each bad smell / Total number of blocks (%) for the deciles D0-D9.

T-student test

After analyzing the distribution and the characteristics of the data set, we found that the main development was more prominent from the decile D2. Therefore, the most effective populations for the study were in D2 and D9, as the final state.

From this point, we investigated the most appropriate procedure to develop the statistical analysis. It was the *t-student* test, which compares the measures obtained -the mean value of bad smells- in two different groups for a given variable, the CT development.

This test allows us to know the probability to obtain results just by chance. If the probability is high, we can say that the results are due to randomness. On the contrary, if the probability is low, we can say that the differences obtained in the test are probably real. In other words, with a low probability we could reject *H0* -both populations do not present differences in the mean value of bad smells.

The variable which we used in the analysis to measure this probability was the *p-value*. By agreement, it is established that a *p-value* less than 0.05 indicates that the result of the analysis is not random and *H0* can be reasonably rejected. Therefore, if the *p-value* is greater than 5%, we cannot affirm that the differences in the results are reliable.

In addition, we calculated the *size effect* for each test. This variable measures the magnitude of the effect studied. In other words, if we find differences in the results, we want to measure how is the size of these differences. A value greater than 0.5 indicates that the *size effect* is 'large' and the differences are significant.

Development of the statistical analysis

After the treatment and analysis of the data set and the choice of the *t-student* test, we developed the study with Jupyter notebook. In particular, we implemented the function *ttest_ind*, which calculates the *t-student* test for the value of two independent groups. For the computation of the *size effect*, we used an automatic online calculator[6].

```
scipy.stats.ttest_ind(population_a, population_b, equal_var=False)
```

[6]https://www.psychometrica.de/effect_size.html

With this function we obtained the *p-value* -we assumed that the populations had no identical variances.

From this point, we carried out the *t-student* test for different populations in order to find reliable results. We describe each case of study as a research question.

RQ1. *T-student* test with deciles

In the first place, we carried out an analysis with two general groups. The first of them was composed of a set of projects in D9 which had a few bad smells in D2. The other one was a set of projects in D9 which had many bad smells in D2.

With this research question we wanted to verify if the population A could present a minor mean value of bad smells in D9 than population B and, in that case, if the differences would be reliable -in terms of *p-value* and *size effect*. In this way, the populations were the following:

- Population A: Projects in D9 which had less than 5 bad smells in D2 (N=168).
- Population B: Projects in D9 which had more than 20 bad smells in D2 (N=95).

RQ2. *T-student* test with deciles for a given CT development

One of the objectives of this analysis was to measure the impact of bad smells in the CT development. In RQ1, we found significant differences in the level of CT between the two groups of study, as we will describe in Chapter 5. For this reason, we repeated the *t-student* test, but based on a specific set of projects of the data set.

We selected those projects whose CT development obtained with Dr. Scratch in D2 was included in the interval [7,12) points of mastery. The objective of this study was to enclose the populations in projects with similar CT development and compare them.

Finally, the total number of this data set was 184 projects and we chose the same populations.

- Population A: Projects in D9 which had less than 5 bad smells in D2 (N=62).
- Population B: Projects in D9 which had more than 20 bad smells in D2 (N=21).

Total samples	Samples in D0	Samples in D1
452	226	226

Table (4.9) Description of the data set used in the statistical analysis without deciles.

RQ3. *T-student* test without deciles

We found very significant results both in RQ1 and RQ2, as we will describe in more detail in Chapter 5. However, we realized that we were comparing very similar projects in each decile and the analysis was not reliable. In addition, we found in the results a very alarming fact: Dr. Scratch tool could assess positively the presence of bad smells.

Therefore, we repeated the *t-student* test with another proposal. In this case, we selected only two snapshots for each project: the first snapshot in which the project got 7 points of CT development, and the last snapshot of the project. We call each situation as D0 and D1, respectively.

- D0: The first snapshot in which the project gets 7 points in the CT development, measured with Dr. Scratch.

- D1: The last snapshot of each project in D0.

The data set extracted consisted of 452 snapshots, divided in 226 snapshots for each population. This distribution is summarized in Table 4.9. Based on these groups, we developed different analysis.

RQ3.1. *T-student* test without deciles for the total number of bad smells

The first research question was carried out with the total number of bad smells, that is, the sum of each type of bad smell. With this analysis, we wanted to verify the same concepts than in RQ1. We selected two populations and developed again the *t-student* test.

- Population A: Projects in D1 which had less than 5 bad smells in D0 (N=129).

- Population B: Projects in D1 which had more than 20 bad smells in D0 (N=11).

RQ3.2. *T-student* test without deciles for default naming

We repeated the *t-student* test for each bad smell independently, in order to avoid the noise introduced by the sum of all of them.

In addition, we selected several cases of study for each analysis by modifying the percentage of the population B. We started with a low percent in case 1 and we increased it in each case. In other words, we wanted to show the differences in the evolution based on different percents of bad smells in population B.

- Case 1
  - (i) Population A: Projects in D1 which had 0% of default names over the total number of sprites in D0 (N=137).
  - (ii) Population B: Projects in D1 which had more than 30% of default names over the total number of sprites in D0 (N=64).
- Case 2
  - (i) Population A: Projects in D1 which had 0% of default names over the total number of sprites in D0 (N=137).
  - (ii) Population B: Projects in D1 which had more than 50% of default names over the total number of sprites in D0 (N=30).
- Case 3
  - (i) Population A: Projects in D1 which had 0% of default names over the total number of sprites in D0 (N=137).
  - (ii) Population B: Projects in D1 which had more than 70% of default names over the total number of sprites in D0 (N=24).
- Case 4
  - (i) Population A: Projects in D1 which had 0% of default names over the total number of sprites in D0 (N=137).
  - (ii) Population B: Projects in D1 which had more than 100% of default names over the total number of sprites in D0 (N=22).

RQ3.3. *T-student* test without deciles for duplicated code

The data set composed of projects with duplicated code was too small. For that reason, we only selected one case of study with the following populations.

- Population A: Projects in D1 which had not duplicated code in D0 (N=217).
- Population B: Projects in D1 which had at least 1 duplicated code in D0 (N=9).

RQ3.4. *T-student* test without deciles for dead code

For the statistical analysis of the dead code, we selected three different cases of study, described below.

- Case 1
  (i) Population A: Projects in D1 which had 0% of dead code over the total blocks in D0 (N=116).
  (ii) Population B: Projects in D1 which had more than 20% of dead code over the total blocks in D0 (N=73).
- Case 2
  (i) Population A: Projects in D1 which had 0% of dead code over the total blocks in D0 (N=116).
  (ii) Population B: Projects in D1 which had more than 30% of dead code over the total blocks in D0 (N=55).
- Case 3
  (i) Population A: Projects in D1 which had 0% of dead code over the total blocks in D0 (N=116).
  (ii) Population B: Projects in D1 which had more than 50% of dead code over the total blocks in D0 (N=28).

RQ3.5. *T-student* test without deciles for attribute initialization

Finally, the last research question was carried out for the attribute initialization. In this case, we selected two different situations for the analysis.

- Case 1

  (i) Population A: Projects in D1 which had 0% of attribute initialization over the total blocks in D0 (N=68).

  (ii) Population B: Projects in D1 which had more than 20% of attribute initialization over the total blocks in D0 (N=15).

- Case 2

  (i) Population A: Projects in D1 which had 0% of attribute initialization over the total blocks in D0 (N=68).

  (ii) Population B: Projects in D1 which had more than 30% of attribute initialization over the total blocks in D0 (N=6).

4.4 Bad smells model

After the development of the full analysis, we discovered that the presence of bad smells in the Scratch projects was very significant, as well as its effect on the CT development. All the results will be detailed in Chapter 5.

From this point, we considered necessary to develop a new environment in the Dr. Scratch tool, in order to show in the first place the results about bad smells and raise awareness about them. We will call this new environment bad smells model.

As we described in Section 1.3, the dashboard of bad smells in the Dr. Scratch interface was very small and hard to interpret. Therefore, the main objective of this phase was to replace the dashboards, shown in Figure 1.4, by the new model with more visual information about bad smells, shown in Figure 1.5. The process of developing the bad smells model was composed of different phases.

4.4.1 *Scratchblocks* library

The dashboard of bad smells showed the information with text format. In this way, it was more difficult to understand the presence of bad smells in the projects and, consequently, to interpret this information and correct them. Therefore, the starting point was to replace the dashboard with the text by visual blocks.

The *scratchblocks* library[7] makes pictures of Scratch blocks from text. We included this library in the code of Dr. Scratch and created a new template with four dashboards, one for each bad smell. In addition, we programmed a new script, *sb3_blocks_mapper.py*, to map the name of the blocks from the JSON file to the *scratchblocks* library.

From this point, we found the problem that the scripts developed in the Phase 1 of this project returned the information about bad smells in a format which *sb3_blocks_mapper.py* could not interpret. Therefore, we had to change some functionalities in each script.

4.4.2 Improvement of the scripts

The scripts which we developed in the Phase 1 of the project were designed to return the information in a text list. For this reason, we had to change the code of all of them and to adapt the list of results of each bad smell to the new format.

In addition, in order to show the results in an easier way to interpret, we added new functionalities, detailed below:

- Sprite naming: detection of default naming for more languages (Personatge, Figura, O actor, Personaia).
- Backdrop naming: detection of default naming for more languages (Fons, Atzeko oihala).
- Dead code: differentiation of dead code for each sprite, shown in Figure 4.7.
- Duplicated code: differentiation and identification of the sprites who have the duplicated blocks, shown in Figure 4.8.

[7] https://github.com/scratchblocks/scratchblocks

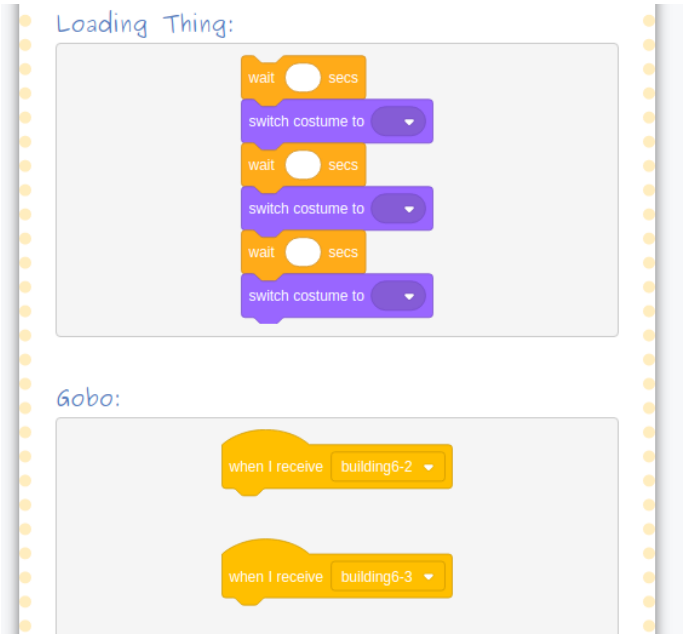


Figure (4.7) Example of the dead code dashboard for different sprites in the bad smells mode.

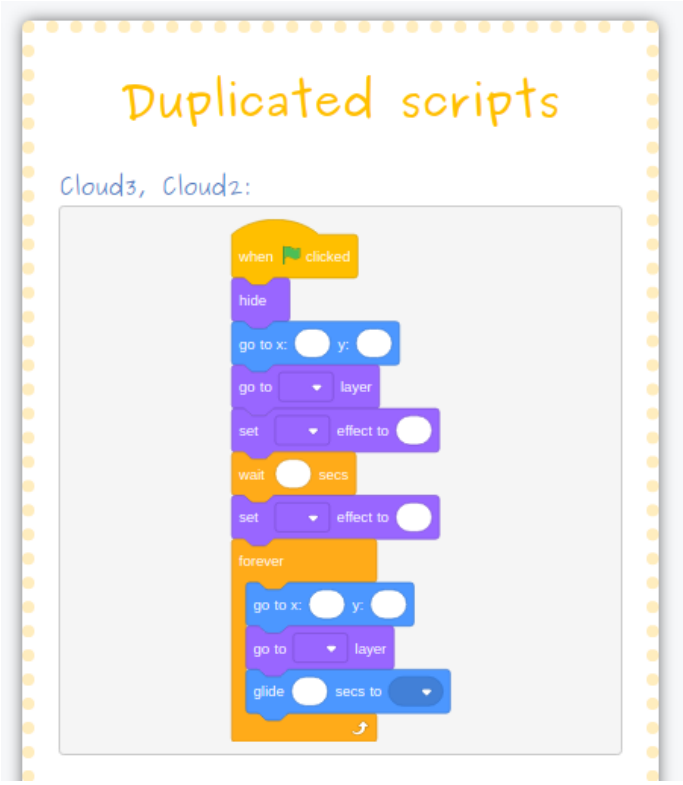


Figure (4.8) Example of the duplicated code dashboard for two sprites in the bad smells mode.

4.4.3 Integration of the bad smells model in Dr. Scratch

The last step for the development of the bad smells model was its integration in the Dr. Scratch code. When users analyzed a Scratch project with Dr. Scratch, they received the response with all the dashboards. We replaced this template for the template with the new model, which can be observed in Figure 4.9.

In addition, we added a new button to access the old dashboards. In other words, once that users have analyzed their results about bad smells, they can navigate to the rest of dashboards with the CT development results and the download of the certificate. We can observe this functionality in Figure 4.10.

Finally, when the user profile was basic (from 0 to 7 points), the Dr. Scratch tool did not show the dashboard with the information about bad smells. Users who are beginning do not need all this information. For this reason, we though that it was better to avoid the bad smells model for this type of users. When the total mastery is less than seven points, Dr. Scratch shows the normal dashboards instead of the bad smells model, as we can observe in Figure 4.11.

4.5 Assessment experiment

After the development of the bad smells model and its implementation, we wanted to verify the effectiveness of our study as the last phase of the project. To do this, we designed an assessment experiment in which two teachers -without any knowledge about bad smells- had to analyze six different Scratch projects, with and without bad smells. In addition, we chose projects with different level of the CT skills, measured with the Dr. Scratch tool. The characteristics of the selected Scratch projects are summarized in Table 4.10. The process was based on two phases, as we described in Chapter 1.

- Phase 1: We developed a general form[8] with a brief description about the experiment. Then, we asked two questions for each Scratch project. Finally, we requested them to order the projects from the worst to the best, according to their judgement.

    1. Evaluate the project from 1 to 10 according to your judgement.
    2. Justify your evaluation.

[8] https://docs.google.com/forms/d/1VA-tMHbsrg1wpKmsn_TOv_kghOkPm2vKb69AxG-dZRU

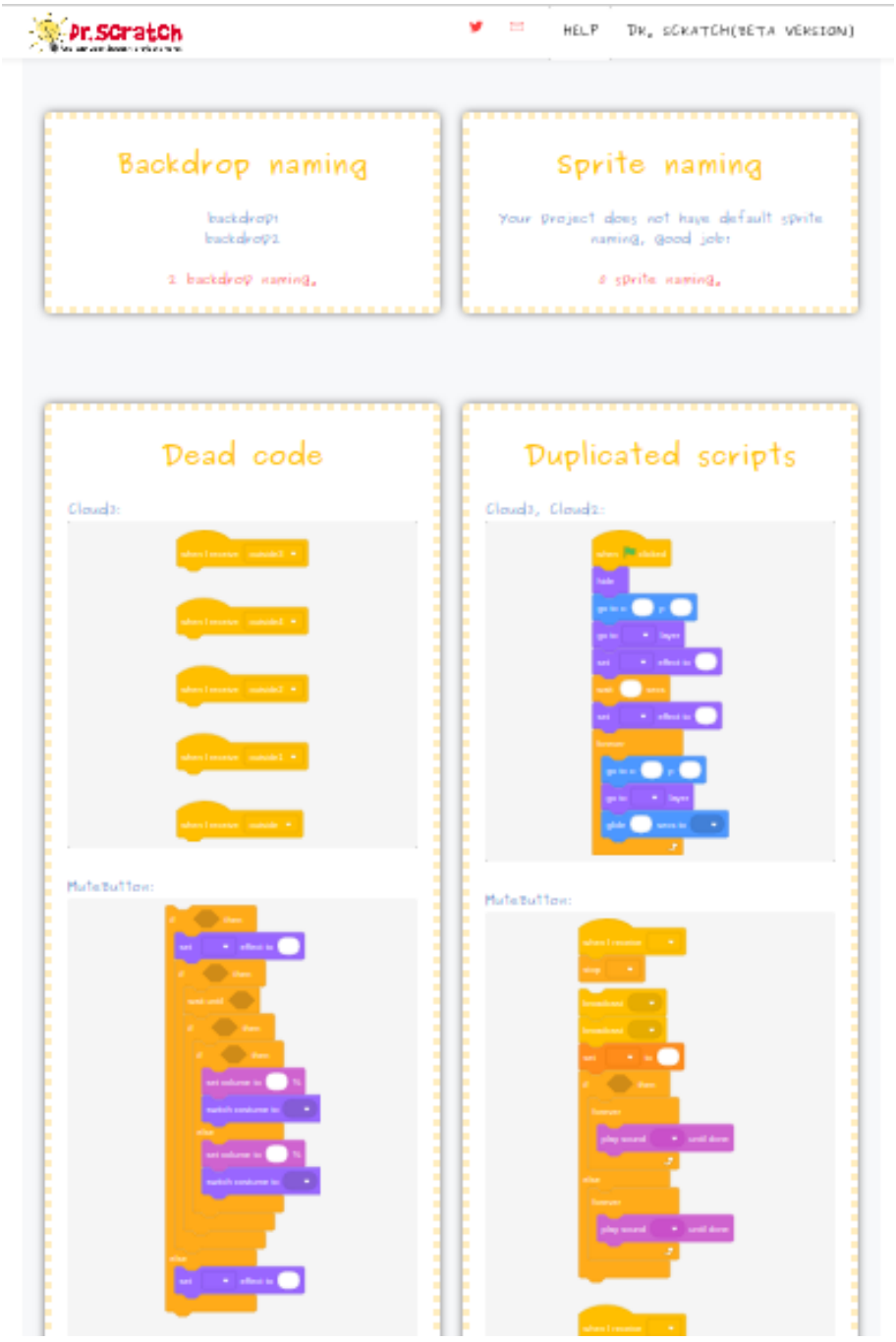


Figure (4.9) New interface of Dr. Scratch for the bad smells model.

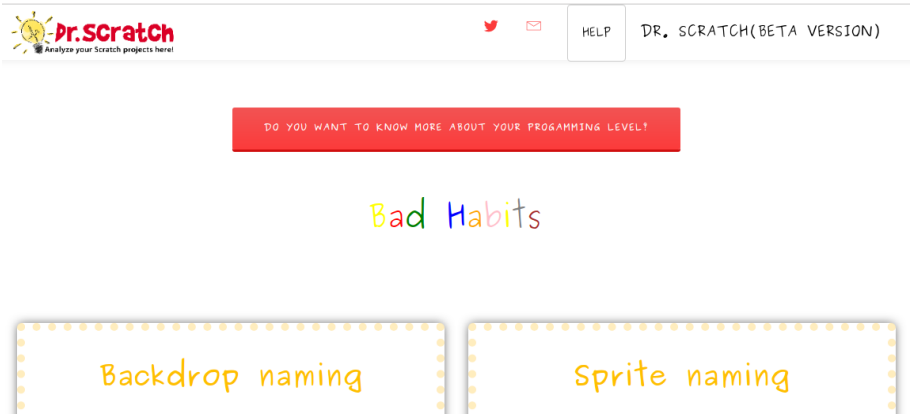


Figure (4.10) Button in the bad smells model to access the rest of the dashboards.

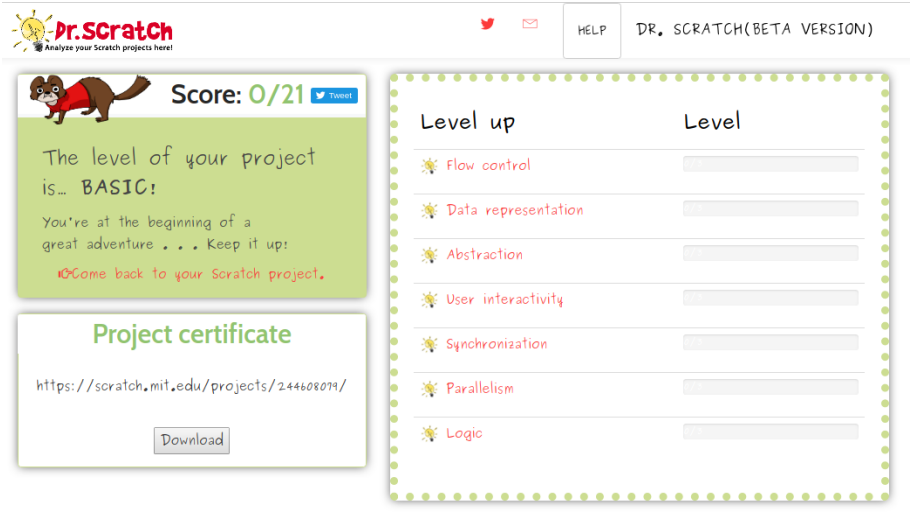


Figure (4.11) Interface of Dr. Scratch for users with Basic profile.

Scratch project	Mastery points	Duplicated code	Dead code	Default sprite naming	Default backdrop naming
Cooking Mama	18	9	7	40	10
Sky Game	17	3	3	14	0
Cutting Down	17	1	0	0	0
Sea Game	12	7	4	11	2
Environment Game	12	1	1	0	0
Maze Game	10	1	1	0	1

Table (4.10) Description of the Scratch projects used in the assessment experiment.

- Phase 2: We developed a more specific form[9] in which we included a brief description about bad smells and explained the main types that could appear in the projects. Then, we asked five questions for each Scratch project. Finally, we requested them to order the projects from the worst to the best, according to their judgement.

    1. Evaluate the presence of repeated code in the project from 1 to 10.
    2. Evaluate the presence of dead code in the project from 1 to 10.
    3. Evaluate the presence of default naming in the project from 1 to 10.
    4. Evaluate the presence of badly initialized attributes in the project from 1 to 10.
    5. Would you modify the evaluation of the project, compared to your evaluation in the part 1 of the experiment?

With this experiment we wanted to verify some aspects. In the first place, that the bad smells are unperceived by the teachers and they are not aware about their presence. In the second place, if the presence of bad smells have an impact in the human assessment once that they are informed about them.

We expect that, when teachers are aware about bad smells, they change their evaluation and penalize the projects with a worse assessment. In this way, we could confirm that Dr. Scratch is analyzing the presence of bad smells wrongly, as we found during this project.

[9] https://docs.google.com/forms/d/1y7bEVjEiY7o0SuNwvD0KCiFCiChBOElr5hBtNXQqT-I/edit?usp=forms_home&ths=true

Chapter 5

Results

This project is composed of different phases, as we have described throughout the dissertation. In this chapter, we will focus on analyzing the results obtained during the bad smells study and afterwards the development of the assessment experiment.

5.1 Analysis of bad smells

In the first place, we will detail the results of the entire study of bad smells, both the general and the statistical analysis. In each section we will try to answer the proposed research questions.

5.1.1 General analysis

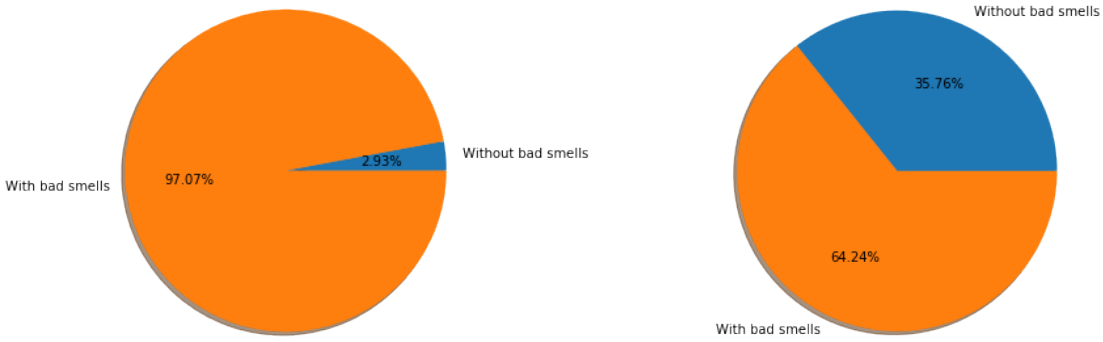
The general preliminary analysis of bad smells, as we described in Chapter 4, is focused on answering different questions related to the impact that bad smells have on Scratch projects. In this section, we detail the results obtained from addressing our research questions using the previously described data set.

RQ1. To what extent are bad smells present in Scratch projects?

As shown in Table 5.1, bad smells can be found in almost all projects in our data set a -over 97% of the projects have at least one bad smell. We expected a high share of projects having bad smells, but this result was a surprise for us, as the presence of bad smells is not only more frequent than expected, but almost general.

	Projects with at least one bad smell	Projects with no bad smells	Total projects
Data set a	97.97% 58,162	2.93% 1,754	100% 59,916
Data set b	64.24% 148,412	35.76% 82,612	100% 231,024

Table (5.1) Presence of bad smells in Scratch projects (RQ1).



(a) Data set a (b) Data set b

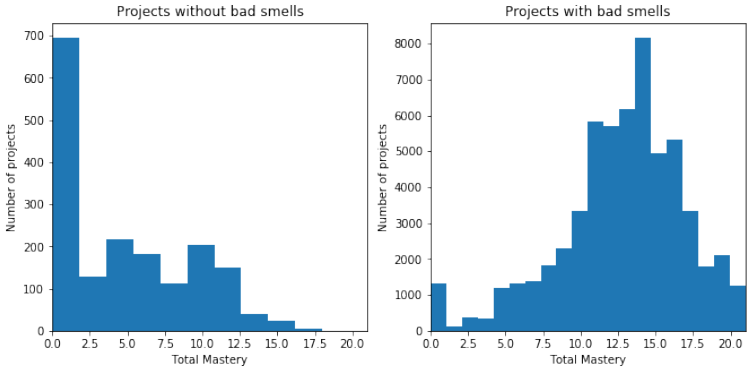
Figure (5.1) Presence of bad smells in Scratch projects (RQ1).

When we repeated the analysis with a larger set of projects, data set b, we found more moderate results. As we can also observe in Table 5.1, around 64% of the projects have at least one type of bad smell. These results are summarized in a more visual way in Figure 5.1. With RQ2, below, we will show why the datasets show a different behavior.

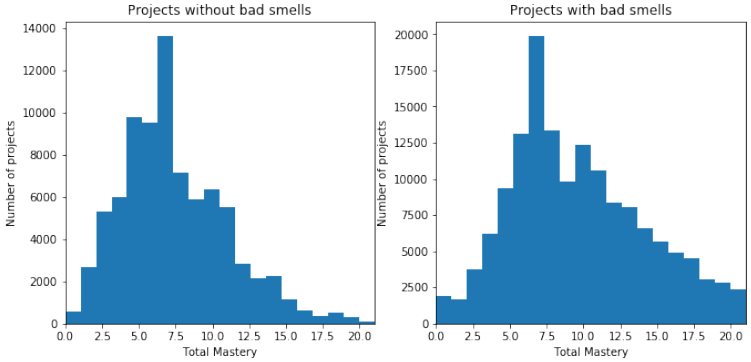
RQ2. Does the development of CT skills relate to a minor presence of bad smells?

We have analyzed in more detail those projects that have and do not have bad smells. In particular, we want to see how the complexity of the projects is related to having a bad smell. We use the mastery required to create a project, as measured by Dr. Scratch, as a proxy for the complexity of the project.

The distribution of each set of projects is shown in Figure 5.2. In Figure 5.2a we can observe that more than 50% of those projects with no bad smells have a total mastery of 0. In other words, these are skeleton projects without any content. The amount of projects with content and



(a) Data set a



(b) Data set b

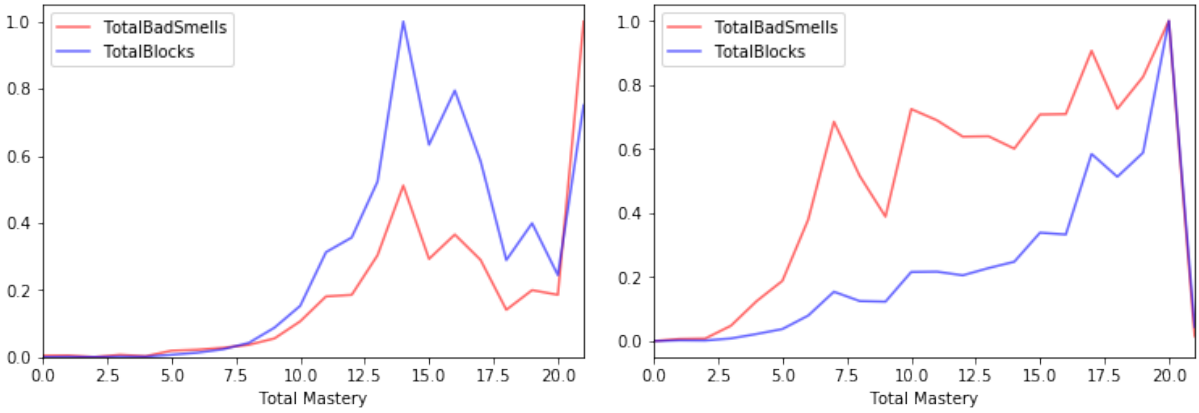
Figure (5.2) Distribution for projects without bad smells and with bad smells (RQ2).

without any bad smell is therefore even lower than calculated in the previous research question. In addition, we see that projects with no bad smells are in the lower part of the complexity ladder.

In Figure 5.2b we can observe, as in the previous question, how the results are more moderate for data set b. In this second analysis, the mean value of the mastery obtained from the set of projects without any bad smells is higher, with 7 points on average (instead of 0 points). This result indicates that projects without bad smells are not mainly empty, although they are still very simple projects.

On the other hand, the results for the set of projects with bad smells are also softened. The distribution moves to the left in the graph and the average mastery decreases, although it continues in the higher part of the complexity ladder.

In summary, bad smells in Scratch can be found in almost all projects. Only a small set of projects with low complexity do not have them.



(a) Data set a

(b) Data set b

Figure (5.3) Relation between the total number of bad smells and the total number of blocks for each mastery level (RQ3).

RQ3. Do projects with more blocks have a higher number of bad smells?

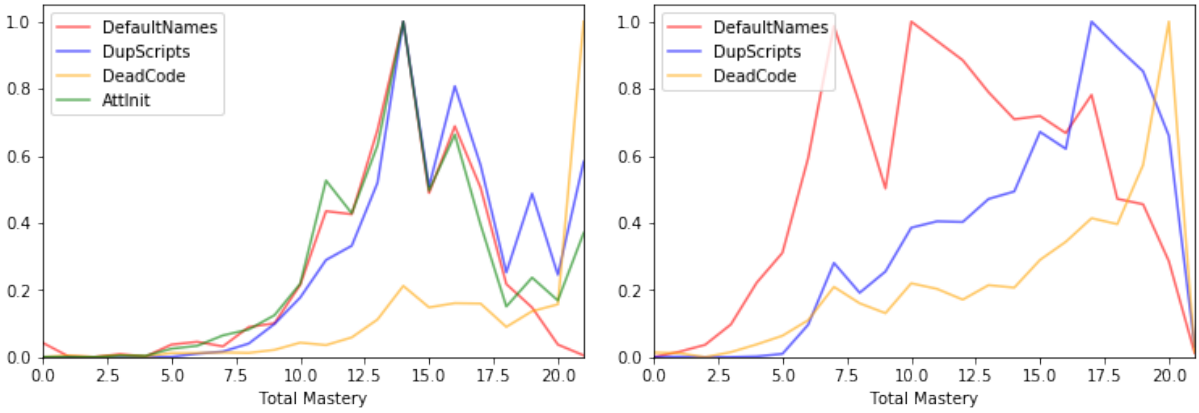
One could argue that projects with more complexity (those with higher values of CT score in Dr. Scratch) usually have more blocks, and that the impact of a code smell there is lower than in less complex projects.

In other words, even if more complex projects have bad smells, their presence is mitigated by the fact that these are large projects. This would imply that achieving high values of CT development means to have less bad smells.

Figure 5.3 visually shows the number of blocks for all projects for a given CT score (blue line) and the number of bad smells in those projects (red line). Both curves have been normalized to their maximum values.

In Figure 5.3a, we can observe how the two curves run almost in parallel. Up to 8 points they share the same ratio, then they share a ratio of around 0.5 up to 21 points, where the ratio is over 1.

Figure 5.3b not only represents the increase of bad smells with the number of blocks, but also a ratio over 1 in almost all the scores of CT (from 1 up to 20), when we normalize both variables. This result implies that bad smells are present in Scratch projects at all levels to a large extent.



(a) Data set a

(b) Data set b

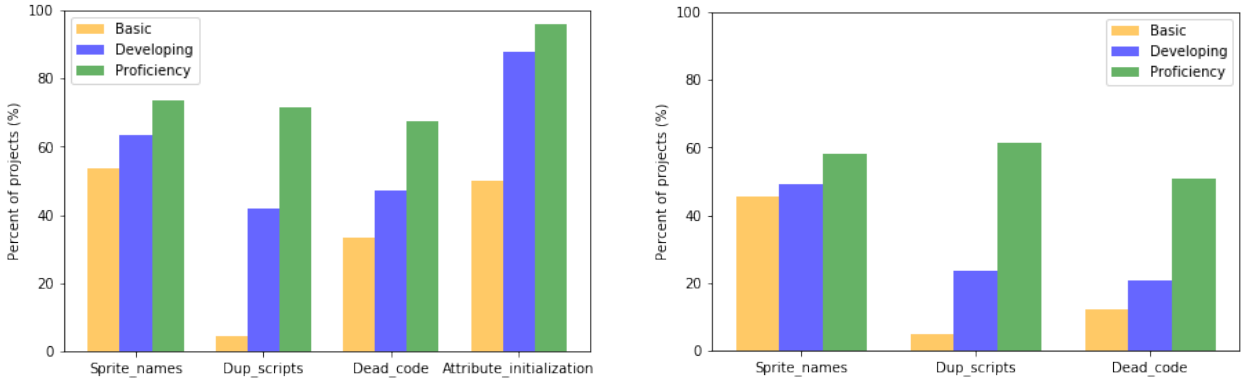
Figure (5.4) Evolution of the different types of bad smells throughout the CT development (RQ4).

RQ4. Can we find a relation among specific bad smells?

From Figure 5.4a (again, this graph is normalized), we can observe that the four different types of bad smells that we have studied have a similar distribution below the proficiency level. The presence of bad smells has a similar behavior for projects up to 17 points of mastery. Then, when the mastery is above 17 points, the behavior of each of them is different: While duplicated scripts and bad attribute initialization continue to slightly grow, the number of dead code blocks grows more abrupt way. However, the number of default names decreases considerably. This last trend may be because in projects with a high number of blocks and objects it is more difficult to program with the default names (Sprite1, Sprite2, ...) instead of personalizing them.

In Figure 5.4b we have only data about three types of bad smells. For data set b, the information about attribute initialization was null, so we cannot represent it.

Regarding the distribution of the other bad smells, all of them grow with a higher CT score, although default naming presents a most extreme curve. The main difference with Figure 5.4a can be observed for proficiency levels. When the mastery is above 17 points, all bad smells decrease considerably (for dead code this effect appears above 20 points). This trend would imply that proficiency profiles have greater awareness about the use of bad smells and they try to mitigate their use.



(a) Data set a

(b) Data set b

Figure (5.5) Presence of bad smells for each proficiency level (RQ5).

RQ5. To which extent can bad smells be identified in each of the CT development phases?

As already seen wit RQ3, the number of bad smells is higher when the total mastery increases. In Figure 5.5, we have represented the percentage of projects that have at least a specific type of bad smell for each level of CT proficiency. As we explained in RQ4, we do not have data about attribute initialization. For this reason, Figure 5.5b only shows three types of bad smells.

In both data set, the percentage of projects from users with basic level that have bad smells is much smaller than the percentage of projects that require a proficiency level. Although Figure 5.5b presents more moderate results than Figure 5.5a, both trends indicate that all bad smells have an incremental evolution with the increase of CT efficiency.

Therefore, it seems that bad smells appear in early phases and instead of disappearing with the development of more advanced CT skills, they become more prominent.

5.1.2 Statistical analysis

The statistical analysis of bad smells is focused in a more specific study about the impact which bad smells have in our variable of interest, CT development. In this section, we describe the results obtained from the proposed research questions.

RQ1. *T-student* test with deciles

In the first place, we analyzed the impact that bad smells have in the CT development in a general way.

Bad Smell Type	Mean A	Mean B	p-value	Size effect
Default name	1.45	8.56	**1.105e-7**	-0.735
Duplicated code	1.18	2.77	**0.000200**	-0.494
Dead code	3.45	33.52	0.054111	-0.250
Attribute initialization	4.23	12.68	**5.576e-7**	-0.685
Total bad smells	10.32	57.53	**0.005191**	-0.367

Table (5.2) Results for the *t-student* test with deciles (RQ1).

	Population A	Population B
D2	9.57	13.52
D9	13.64	14.73

Table (5.3) Mean value of the CT development scored with Dr. Scratch in D2 and D9 (RQ1).

– Population A: Projects in D9 which had less than 5 bad smells in D2 (N=168).

– Population B: Projects in D9 which had more than 20 bad smells in D2 (N=95).

We found very interesting results. Except for *dead code*, the results are statistically significant (i.e., the *p-value* of all of them is below 0.05, which implies that the differences between 'Mean A' and 'Mean B' are significant and not random). In other words, we found a different evolution between both populations and we could confirm that projects with a few bad smells in the beginning of a learning process, will have less bad smells in a period of time than projects which started with a higher value of bad smells.

In addition, the *size effect* presents a value about 0.5 or greater for all of them, except for the total number of bad smells. This may be caused by the noise introduced in the sum of all of them. These results are summarized in Table 5.2.

However, we can observe in Table 5.3 how the mean value of CT development in D2, measured with Dr. Scratch, is 9.57 for population A and 13.52 for population B. The different level of the CT skills at the beginning of a learning process could influence in the evolution of both groups. For this reason, we repeated the analysis for a more specific population with a more enclosed level of the CT skills.

Bad Smell Type	Mean A	Mean B	p-value	Size effect
Default name	1.34	7.90	**0.001452**	-0.923
Duplicated code	0.90	1.95	0.062411	-0.494
Dead code	3.16	10.38	0.136856	-0.390
Attribute initialization	3.97	11.86	**0.020254**	-0.634
Total bad smells	9.37	32.10	**0.000208**	-1.107

Table (5.4) Results for the *t-student* test with deciles for a given CT development (RQ2).

	Population A	Population B
D2	9.45	9.81
D9	12.90	13.14

Table (5.5) Mean value of the CT development scored with Dr. Scratch in D2 and D9 (RQ2).

RQ2. *T-student* test with deciles for a given CT development

As we described in Chapter 4, we repeated the analysis with projects whose CT level was included in the interval [7,12).

– Population A: Projects in D9 which had less than 5 bad smells in D2 (N=62).

– Population B: Projects in D9 which had more than 20 bad smells in D2 (N=21).

From this point, we obtained slightly worse results. Regarding *duplicated code*, the *p-value* obtained is greater than 0.05, so we cannot confirm that the differences between the mean value of bad smells in population A and population B are significant and not random. The results are summarized in Table 5.4.

On the other hand, the mean values of CT development in D2 are similar for each group, with a value of 9.45 points for population A and 9.81 points for population B, as we can observe in Table 5.5. However, the evolution of CT skills in D9 is not as we expected. Not only are the values very similar; they are slightly higher for population B. This result could imply that Dr. Scratch assesses bad smells positively instead of penalizing them.

In order to verify this hypothesis, we repeated the analysis with other populations and without deciles.

Bad Smell Type	Mean A	Mean B	p-value	Size effect
Default name	3.02	1.45	0.097957	0.531
Duplicated code	1.44	1.23	0.662653	0.139
Dead code	4.99	16.18	0.209101	-0.421
Attribute initialization	5.42	8.88	0.499307	-0.220
Total bad smells	14.88	27.73	0.191010	-0.438

Table (5.6) Results for the *t-student* test without deciles for the total number of bad smells (RQ3.1).

	Population A	Population B
D0	7.0	7.0
D1	13.15	13.45

Table (5.7) Mean value of the CT development scored with Dr. Scratch in D0 and D1 (RQ3.1).

RQ3.1. *T-student* test without deciles for the total number of bad smells

The first analysis was developed in a general way for all bad smells.

– Population A: Projects in D1 which had less than 5 bad smells in D0 (N=129).

– Population B: Projects in D1 which had more than 20 bad smells in D0 (N=11).

The results were not significant, since the *p-value* and the *size effect* were not relevant in any bad smell, as we can observe in Table 5.6. We thought that these bad results were due to the noise introduced in the sum and the general analysis of each bad smell. For these reasons, we developed more specifically the *t-student* test for each type of bad smell.

On the other hand, nor did we find significant differences in the evaluation of the CT score by Dr. Scratch. As it is shown in Table 5.7, the mean value of CT development in D1 is so close for population A and population B. Therefore, we could not confirm the hypothesis found in RQ2.

Default Name	Median A	Median B	p-value	Size effect
Case 1	0.0	3.0	**0.001889**	-0.486
Case 2	0.0	6.0	**0.004488**	-0.617
Case 3	0.0	6.0	**0.000217**	-0.934
Case 4	0.0	6.0	**0.000469**	-0.919

Table (5.8) Results for the *t-student* test without deciles for default naming (RQ3.2).

RQ3.2. *T-student* test without deciles for default naming

In the first place, we realized that the *median* as measure variable was more relevant than the *mean*. For this reason, since this analysis, we used *median* instead of *mean*.

In the second place, we found very interesting results. The *p-value* for all the study cases was below 0.05, which indicates that the differences between population A and population B are relevant. In addition, the *size effect* had also a good value, close to 0.5 for case 1 and greater for the rest of them. These results are summarized in Table 5.8.

However, we confirmed our hypothesis in RQ2. As we can observe in Table 5.9, the CT score measured by Dr. Scratch in D1 increases with each case -that is, with the increase of the percent of default names-, due to the higher number of blocks and sprites. This result means that Dr. Scratch does not take into account the increase in default names, but only the increase in the number of blocks and sprites.

RQ3.3. *T-student* test without deciles for duplicated code

The results of this analysis are not as relevant as the results for default naming, as we can observe in Table 5.10. The *p-value* and the *size effect* do not indicate significant results and we cannot confirm that the presence of duplicated code in the beginning of a learning process impacts in the CT development.

However, our hypothesis can be confirmed again. In Table 5.11 we can observe how the CT score for population B in D1 is higher than the CT score for population A. This result indicates that Dr. Scratch is assessing positively the presence of duplicated code by the simple fact that the project has a larger number of blocks.

		Median CT score		Median blocks		Median sprites	
		A	B	A	B	A	B
D0	Case 1	7.0	7.0	19.0	14.0	2.0	3.0
	Case 2	7.0	7.0	19.0	11.5	2.0	2.5
	Case 3	7.0	7.0	19.0	11.5	2.0	2.0
	Case 4	7.0	7.0	19.0	12.0	2.0	2.0
D1	Case 1	12.0	13.5	116.0	143.0	7.0	9.0
	Case 2	12.0	14.0	116.0	180.5	7.0	9.5
	Case 3	12.0	14.0	116.0	180.5	7.0	9.5
	Case 4	12.0	14.0	116.0	187.5	7.0	9.5

Table (5.9) Median value of CT development scored with Dr. Scratch, blocks and sprites for default naming (RQ3.2).

Duplicated Code	**Median A**	**Median B**	**p-value**	**Size effect**
Case 1	0.0	1.0	0.217311	-0.455

Table (5.10) Results for the *t-student* test without deciles for duplicated code (RQ3.3).

	Median CT score		**Median blocks**	
	A	B	A	B
D0	7.0	7.0	17.0	39.0
D1	13.0	14.0	131.0	218.0

Table (5.11) Median value of blocks and CT development scored with Dr. Scratch for duplicated code (RQ3.3).

Dead Code	Median A	Median B	p-value	Size effect
Case 1	0.0	5.0	0.096581	-0.250
Case 2	0.0	6.0	0.080498	-0.289
Case 3	0.0	5.0	0.131713	-0.325

Table (5.12) Results for the *t-student* test without deciles for dead code (RQ3.4).

		Median CT score		Median blocks	
		A	B	A	B
D0	Case 1	7.0	7.0	13.5	21.0
	Case 2	7.0	7.0	13.5	21.0
	Case 3	7.0	7.0	13.5	22.5
D1	Case 1	14.0	8.0	198.5	58.0
	Case 2	14.0	7.0	198.5	31.0
	Case 3	14.0	8.5	198.5	71.5

Table (5.13) Median value of blocks and CT development scored with Dr. Scratch for dead code (RQ3.4).

RQ3.4. *T-student* test without deciles for dead code

The results of the *t-student* test for dead code are very similar to the analysis of duplicated code. The *p-value* and the *size effect* have not significant values. Therefore, nor can we confirm that the presence of dead code in the beginning of a learning process has an impact in the CT development. These results are summarized in Table 5.12.

Regarding the assessment of Dr. Scratch, we found the same effect in this analysis. As we can observe in Table 5.13, the CT score in D1 is higher for population A than the CT score for population B. However, this result is not due to the presence or not of dead code, but to the larger number of blocks in population A. When the number of blocks in population B decreases -maybe because of the elimination of dead code- the value of the CT development also decreases, instead of receiving a positive recognition.

Attribute Initialization	Median A	Median B	p-value	Size effect
Case 1	3.0	6.0	0.101538	-0.499
Case 2	3.0	10.0	0.092452	-0.873

Table (5.14) Results for the *t-student* test without deciles for attribute initialization (RQ3.5).

		Median CT score		Median blocks	
		A	B	A	B
D0	Case 1	7.0	7.0	14.0	9.0
	Case 2	7.0	7.0	10.0	7.0
D1	Case 1	14.0	14.0	175.5	189.0
	Case 2	14.0	14.5	175.5	328.5

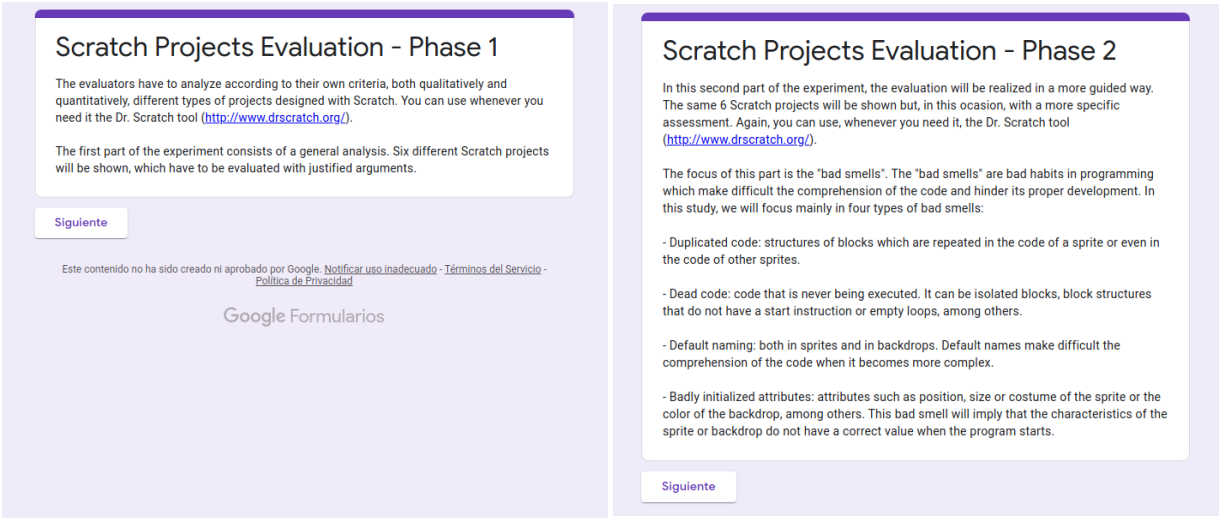
Table (5.15) Median value of blocks and CT development scored with Dr. Scratch for attribute initialization (RQ3.5).

RQ3.5. *T-student* test without deciles for attribute initialization

Lastly, we obtained the same effects on the results of the analysis with attribute initialization. In Table 5.14 we can observe that the values of *p-value* and *size effect* are not relevant. Therefore, we cannot confirm that a worse initialization of attributes has an impact in the CT development process.

In addition, we can observe slightly the trend of the previous analyzes in Table 5.15. The value of CT scored with Dr. Scratch is a little greater, from 14.0 to 14.5 points, when the number of blocks increases significantly, from 175.5 to 328.5.

In summary, with a more specific analysis with the *t-student* test and more controlled populations, we cannot affirm that the presence of bad smells has an impact in the development of the CT skills in a learning process. However, we have found an alarming fact that should be analyzed in more detail: not only the Dr. Scratch tool does not assess negatively the presence of bad smells in the Scratch projects, but gives them more punctuation if they are present in projects with a larger number of blocks.



(a) Phase 1 (b) Phase 2

Figure (5.6) Introduction sections of the forms used in the assessment experiment.

5.2 Assessment experiment

The assessment experiment is a tool to study in a more detailed way the problems found in the functionality of Dr. Scratch during the analysis of bad smells.

As we described in Chapter 4, we divided the experiment into two different parts. In the first phase, we asked both teachers a general evaluation of the Scratch projects, without any specific instruction or guide. In the second phase, we asked them to redo the same evaluation, but with an explicit description about bad smells and guided questions about them. We expected a significant change in their assessments. In this way, we could analyze whether the behaviour which teachers develop is the same which we expected from Dr. Scratch in the assessment of bad smells.

In Figure 5.6 we can observe the introduction section of the forms for each phase, respectively. The questions of each part, as well as a brief code description of the Scratch projects, are described in a more detailed way in the Appendix A.

5.2.1 First phase

In the first place, we asked both teachers to evaluate six Scratch projects in a general way and to order them after the analysis. The obtained results are summarized in Table 5.16, where the ‘Assessment experiment’ column contains the evaluation of each teacher -the first score belongs

Scratch project	Mastery points	Dup. code	Dead code	Sprite naming	Attribute initialization	Assessment experiment
Cooking Mama	18	7	7	40	19	4 | 4
Sky Game	17	4	3	14	44	7 | 9
Cutting Down	17	1	0	0	1	8 | 7
Sea Game	12	6	4	11	6	2 | 5
Environment Game	12	1	1	0	0	4 | 4
Maze Game	10	1	1	0	2	2 | 2

Table (5.16) Results of the assessment in the first phase of the experiment.

to the assessment of the first teacher and the second one, to the second teacher.

Cooking Mama project

In the case of the 'Cooking Mama' project, both teachers evaluated it with four out of ten points. Some of the justifications they gave were: 'Too many objects', 'difficulty understanding the code', 'although the program should have more points, I give it a low score because there are numerous generic names and make difficult the legibility of the code'. Therefore, both teachers were able to detect the presence of default sprite naming and they scored the project negatively.

On the contrary, Dr. Scratch assesses 'Cooking Mama' with 18/21 points of mastery, evaluating only the richness of blocks that the project has, without taking into account the presence of bad smells.

Sky Game project

Regarding the 'Sky Game' project, both teachers evaluated it with higher marks than the previous one. Some of their comments were: 'It is a complete and elaborate program, although it has some repeated blocks and the objects should have more specific names', 'very elaborate code although it is not very well understood how it works'. Therefore, both of them detected less amount of bad smells and gave to the project a better evaluation than 'Cooking Mama'.

In this case, we have the same effect on the assessment of Dr. Scratch: high mastery -17/21 points- without taking into account the presence of bad smells in the evaluation.

Cutting Down project

Despite this project has a few presence of bad smells, both teachers coincided in the same argument: 'very simple code, with too many repeated elements'. Again, they detected the presence of duplicated code and penalized partially the project. In this case, although is a simple project with a basic code, Dr. Scratch assesses it with a high punctuation due to the presence of duplicated blocks.

Sea Game project

The evaluation of 'Sea Game' was very low, because it is a project with both bad smells and a basic level of CT development. Teachers commented aspects of the code, such as dead blocks, 'there are objects without any functionality' or duplicated code, 'there are repeated code (for example, sprite5, sprite8 and sprite9 all order the Ocean Acid variable to be displayed... with doing it once would suffice)'.

Environment Game project

Regarding 'Environment Game', both teachers considered it as a basic project giving it a low score. Due to the small amount of bad smells, they justified their assessment based on code failures: 'The synchronization between objects is not appropriate', 'sending unnecessary messages', 'unnecessary parallelism, it could be done with two conditions within the same loop'.

Therefore, in this case in which the presence of bad smells is not very relevant, the evaluations of the teachers -4/10 points- are in-line with the assessment of Dr. Scratch -12/21 points.

Maze Game project

Finally, we obtained the same effect in this project. As it is a project without a lot of bad smells, both teachers focused their evaluation in the code. For this reason, both teachers agreed with the assessment of Dr. Scratch and gave to the project the worst evaluation. They underlined aspects such as the disorganization or the lack of coherence in the code and scored it with only two out of ten points.

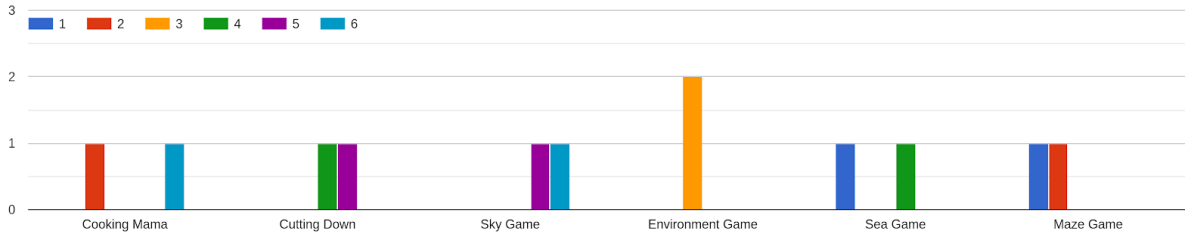


Figure (5.7) Projects order in the first phase of the assessment experiment.

Ordering projects by their development of CT skills

In the last question, we obtained different evaluations, as we can observe in Figure 5.7. However, teachers coincided in some aspects: They considered 'Maze Game' among the worst projects, 'Environment Game' in the middle of the ranking and 'Cutting Down' and 'Sky Game' among the best projects.

The projects order is mainly based on the level of CT development considered by the teachers. However, we can appreciate how the presence of bad smells has influenced in their classification.

In summary, we could conclude with the first phase of the experiment that teachers penalize the presence of bad smells, even without being informed about them. Therefore, when they evaluate projects without many bad smells, they coincide with the assessment of Dr. Scratch. On the contrary, when the projects have a lot of bad smells, they detect them and analyze the projects negatively, in contrast to the assessment of Dr. Scratch.

Therefore, both teachers developed the expected behaviour in the assessment of bad smells even in the first phase of the experiment.

5.2.2 Second phase

In the second phase of the experiment, we asked both teachers to evaluate the same six Scratch projects again, but with a previous explanation about bad smells and more guided questions. In this way, they evaluated each bad smell separately from 0 to 10 points in each project, as they considered.

Duplicated Code			
Scratch project	**Dr. Scratch**	**Teacher A**	**Teacher B**
Cooking Mama	7	1	4
Sky Game	4	5	8
Cutting Down	1	3	5
Sea Game	6	7	6
Environment Game	1	1	10
Maze Game	1	8	1

Table (5.17) Results of the duplicated code assessment in the second phase of the experiment.

Finally, we requested them to order the projects again, in order to verify whether the knowledge about bad smells could impact in their evaluations. The results obtained in the assessment of each type of bad smell are described below.

Evaluate the presence of repeated code in the project

The answers received about the repeated code are summarized in Table 5.17. The column which contains the assessment of Dr. Scratch describes the number of duplicated scripts found in the code of the Scratch projects. The columns 'Teacher A' and 'Teacher B' include the evaluations carried out by the teachers respectively, from 0 to 10 points.

We can observe the variety of results even for the same project. For instance, in the case of 'Environment Game', *Teacher B* considers that duplicated code has the maximum presence in the code, while *Teacher A* opines the opposite -taking into account that they gave the same punctuation to the project in the first phase of the experiment.

Therefore, we could affirm that the evaluation of the impact which this bad smell has on the projects is subjective and depends on the perception of each person.

Evaluate the presence of dead code in the project

The results obtained in the evaluation of dead code are summarized in Table 5.18. We can observe the same effect described previously, even more prominent. The results vary much, even being opposed in projects such as 'Environment Game' or 'Cutting Down'.

Dead Code			
Scratch project	**Dr. Scratch**	**Teacher A**	**Teacher B**
Cooking Mama	7	2	4
Sky Game	3	1	8
Cutting Down	0	1	10
Sea Game	4	9	6
Environment Game	1	1	10
Maze Game	1	3	8

Table (5.18) Results of the dead code assessment in the second phase of the experiment.

On the other hand, we could consider the possibility of wrong evaluations from teachers, by introducing erroneous values. That fact could explain, for instance, the value of 10 points which *Teacher B* gives to 'Cutting Down', taking into account that the project does not have dead blocks.

Evaluate the presence of default naming in the project

As we can observe in Table 5.19, we found the same effect in the results of the presence of default naming in the Scratch projects. Again, we could consider some wrong answers. For instance, *Teacher B* evaluates with 10 points the 'Environment Game' project or both *Teacher B* and *Teacher A* give high scores to 'Maze Game'.

Therefore, we could not affirm that teachers evaluate bad smells in a subjective way and their answers are not wrong.

Evaluate the presence of badly initialized attributes in the project

Finally, in Table 5.20 are summarized the results of the attribute initialization assessment. In this case, we can observe how both teachers give 9 and 10 points to the 'Sky Game' project, which contains the highest number of badly initialized attributes.

On the contrary, we observe possible wrong evaluations in the assessment of the 'Environment Game' project -it does not have badly initialized attributes and both teachers evaluate their presence with high marks.

Default Naming			
Scratch project	**Dr. Scratch**	**Teacher A**	**Teacher B**
Cooking Mama	40	10	1
Sky Game	14	9	3
Cutting Down	0	5	10
Sea Game	11	8	1
Environment Game	0	1	10
Maze Game	0	8	9

Table (5.19) Results of the default naming assessment in the second phase of the experiment.

Attribute Initialization			
Scratch project	**Dr. Scratch**	**Teacher A**	**Teacher B**
Cooking Mama	19	1	2
Sky Game	44	9	10
Cutting Down	1	1	9
Sea Game	6	5	9
Environment Game	0	6	9
Maze Game	2	5	3

Table (5.20) Results of the attribute initialization assessment in the second phase of the experiment.

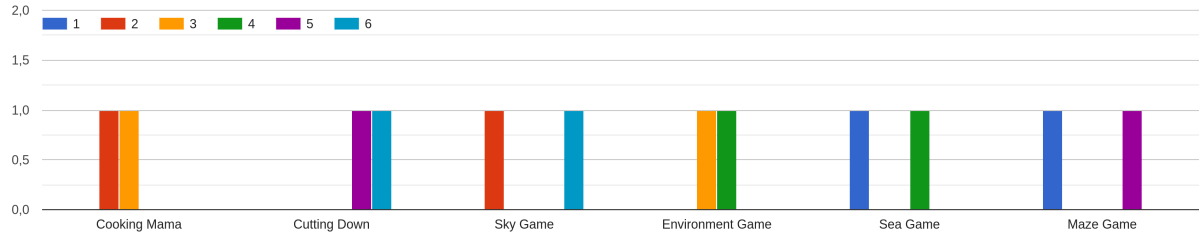


Figure (5.8) Projects order in the second phase of the assessment experiment.

Would you modify the evaluation of the project, compared to your evaluation in the phase 1 of the experiment?

In this question, we expected significant changes in the answers. However, in the first phase of the experiment, teachers evaluated all of the Scratch projects taking into account the presence of bad smells. In this way, they justified their evaluations based on the amount of bad smells, unconsciously.

Therefore, when they received the information about bad smells, they did not change their assessments. Only in the 'Sky Game' project, *Teacher B* commented: 'I would lower the mark a little, I think I gave the project a very high score'.

Ordering projects by their development of CT skills

In the last question, *Teacher B* maintained the same order than in the previous phase. Therefore, the knowledge about bad smells and its specific evaluation did not impact on his initial assessment.

On the contrary, *Teacher A* decided to change the order, penalizing the 'Maze Game' project and improving the 'Cooking Mama' and 'Sky Game' projects, as we can observe in Figure 5.8. After analyzing his answers, the reasons could be very different: an impact on the perception of the projects due to the presence of bad smells, wrong answers as we described previously or difficulty to understand each type of bad smell explained, among others.

Therefore, in this case we could not verify whether the knowledge about bad smells has an impact on his initial assessment.

In summary, throughout the second phase of the experiment we obtained, in some extend, illogical results. The reasons could be the misunderstood of the explanation of each type of bad smell, the description of the questions or, consequently, the wrong answers obtained. Therefore, we did not find the impact that we expected.

However, during the first phase of the experiment we obtained the results that we expected: the impact of bad smells in the assessment of Scratch projects.

Therefore, we should dedicate more time in explanations about bad smells, in order to inform properly about their presence and impact. In this way, we could avoid its unconscious evaluation and raise awareness about their importance.

Chapter 6

Conclusions

In this chapter, the achieved results -based on the proposed objectives-, as well as the encountered problems during the project, are analyzed. In addition, I reflect on the learned and applied knowledge during its development. Lastly, we contemplate possible lines of work to follow, in order to continue and improve the work.

6.1 Achievement of objectives

Regarding Chapter 2, the main objective of this project was composed of four differentiated parts.

In the first place, we wanted to update the Dr. Scratch tool according to the new version of Scratch. This objective has been successfully achieved because we have implemented a stable version in the Google Cloud Platform, which is compatible with the new format of the Scratch projects, analyzing them and showing the same results as the previous version. However, as we described during this dissertation, we could not include the analysis of the badly initialized attributes. In its place, we developed the analysis of default backdrop naming and Dr. Scratch shows this information instead.

In the second place, we wanted to develop a complete study about bad smells, with the main objective of analyzing their impact on the CT development. Thanks to the extensive data set we collected, we developed different types of analyzes and found very interesting results. We have discovered that bad smells are present in most of the Scratch projects, and their presence is unperceived by the programmers. In addition, we extended the analysis in a statistical way, with

the *t-student* test, and found that the Dr. Scratch tool evaluates in a positive way the presence of bad smells, instead of penalizing them. Therefore, we have answered the proposed research questions and achieved this goal.

From the analysis of bad smells, we wanted to design and implement a new model in the web interface of Dr. Scratch, in which bad smells had more importance. We have also achieved this objective. We developed a model with different dashboards in which we showed the four types of bad smells -duplicated code, dead code, default sprite naming and default backdrop naming- in a more visual way. We replaced the list format of the previous version of Dr. Scratch and implemented simpler and bigger dashboards with blocks and visual format. In addition, we hid the dashboards with the CT development and we showed them only when the programmers remove all the bad smells in their projects. In this way, we raise awareness about the presence of bad smells and give them the importance they need.

Lastly, the last part of the general objective was to design an experiment in order to verify the effectiveness of our research. The initial idea was to carried it out with at least 10 different teacher profiles. However, it was complicated to find them and develop it in a short period of time. Finally, we conducted the experiment with two secondary teachers. It was composed of two phases, in which they had to analyze six Scratch projects, first without any knowledge about bad smells and then, with a brief description of them. We have achieved this objective because we found that even though teachers are not aware about the presence of bad smells, contrary to we expected, they evaluated the Scratch projects based on their presence, unconsciously. Therefore, they developed the expected behaviour in the assessment of bad smells.

Regarding the specific objectives, as we described in Chapter 2, they were the intermediate steps to achieve the general objective. Therefore, we can affirm that all of them have been reached.

6.2 Application of learned knowledge

During the development of this project, I have applied numerous skills learned during my degree and master. Due to the variety of its content and objectives, I have needed to use knowledge of very different subjects.

1. *Programming Fundamentals*. This subject was my first contact with the programming area. Thanks to it, I learned the basic concepts of programming, such as loops, conditionals, or abstraction problems, among others.

2. *Programming in Telecommunication Systems*. This subject was the starting point of the programming learning process. I had to develop complicated projects that have helped me to encourage the programming problems found during this work.

3. *Computer Network Architecture*. Thanks to this subject I acquired the knowledge related to communications, learning concepts such as the HTTP protocol or the client-server structure, among others. It has been very necessary to understand and modify the architecture of Dr. Scratch.

4. *Services and Applications in Computer Networks*. My first contact with the Python language was with this subject. In addition, I learned web programming, in particular Django. Without a doubt, this subject has been the most important for the development of the new version of Dr. Scratch.

5. *Telematic Application Development*. This subject was an extension of the previous one. I learned more specific knowledge about web programming, such as CSS, HTML or Bootstrap, among others. For the development of the bad smells model it has been indispensable.

6. *Multimedia Information Processing and Management*. On this subject I learned concepts related to the data treatment, such as data filtering, sample processing or database structures, among others. It has been a key subject for the collection and management of the data set used in the analysis of bad smells.

7. *Network and Services Management and Operation*. With this subject I learned the concepts related to the cloud platforms. Although it was mainly based on Amazon Web Services, it helped me to understand the functionalities of the Azure and Google Cloud platforms.

8. *Projects Management*. Finally, this subject includes the general concepts for the development of the complete project. I learned different skills and tools, such as the Gantt diagram, which have helped me to organize and manage all the work during these two years.

6.3 Learned lessons

Throughout this project, I have dealt with different problems and challenges. Thanks to that, I have increased my knowledge about some of the concepts described above and have learned new ones.

1. I have learned how to work with the cloud platforms, as well as how to manage their main services.

2. I had never worked with the production environment and, throughout this project not only I had to develop a tool -the new version of Dr. Scratch-, but I also had to migrate it between different platforms.

3. I have learnt the management of tools such as Apache and MySQL, and the configuration of their main files.

4. I have increased my knowledge about Django and web programming, as well as the design of web interfaces with CSS and HTML.

5. I have improved my knowledge about data treatment, with bigger and more complicated data set.

6. I have learnt other kind of data analysis, with the statistical *t-student* test, as well as to interpret different statistical variables and results.

7. I have improved my level of English, as well as my ability to write papers and defend them in the conferences, thanks to the participation in different congresses throughout this project.

6.4 Future works

Throughout this research, we have found a very important fact which I consider that is the main future work in order to continue and improve this work: the correction of the functionality of Dr. Scratch, penalizing the presence of bad smells instead of supporting them. This objective is very interesting and necessary, because it would help to raise awareness about bad smells and would support the obtained results in this project.

Despite this task is the most important, there are other future lines of work which could improve considerably this project.

- The development of a script which analyzes the badly initialized attributes. In the new version of Dr. Scratch we could not include it in the functionality, as we described throughout the dissertation.

- The development of scripts which analyze other kind of bad smells such as the length or complexity of the code, and their integration in the code of Dr. Scratch.

- Regarding to the analysis of bad smells, although we carried out an extensive a complete analysis, we could not find the results that we expected. For this reason, another important future line of work is the development of a new statistical analysis with different data set and populations in order to find better results.

- One of the encountered problems was that we carried out the assessment experiment with only two people. Therefore, it would be necessary to repeat it with a higher variety of teacher profiles. In this way, we could verify the effectiveness of our study more reliably.

- To give greater visibility to the new model of bad smells in the interface of Dr. Scratch. During this project we have designed and implemented this model, but it would be necessary to raise awareness about its importance, doing activities such as workshops or talks in schools, among others.

- Lastly, as a more ambitious goal, we would like to include more languages in the Dr. Scratch tool -we included Russian. It would help to create a social community which would improve the number of users which use Dr. Scratch considerably.

Appendix A

Detailed assessment experiment

In this section, we describe more in detail the Scratch projects used during the experiment, as well as their URLs in the Scratch platform. The main objective of showing the code of the projects is to facilitate the understanding, in some extend, the arguments and assessments of the teachers throughout their analysis.

Cooking Mama

In Figure A.1 we can appreciate different bad smells in the code of the *Cooking Mama* project. For instance, we can observe several repeated blocks or default naming both in sprites (Sprite1, Sprite2, Sprite3) and backdrops (Backdrop1, Backdrop2, ..., Backdrop10). The project is available at `https://scratch.mit.edu/projects/366067852/`.

Figure (A.1) *Cooking Mama* project.

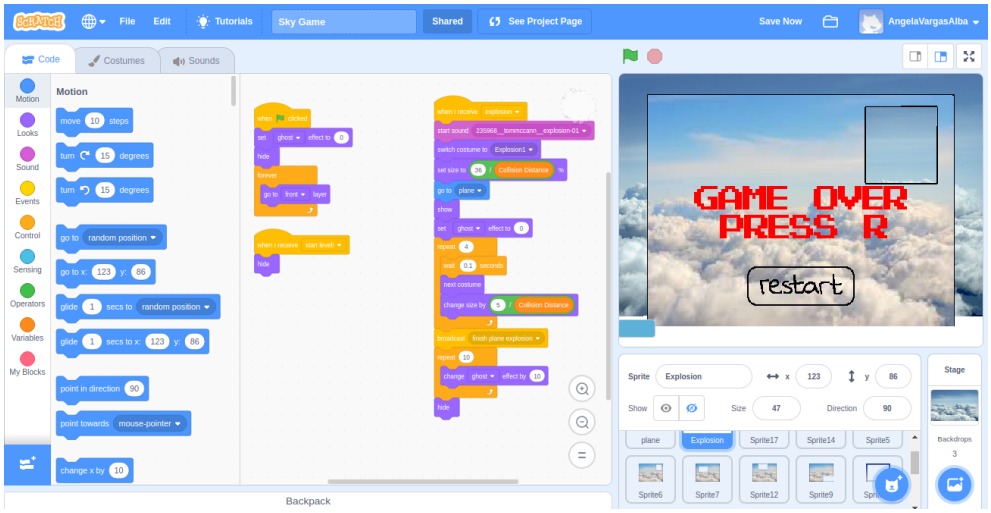


Figure (A.2) *Sky Game* project.

Sky Game

We can appreciate a simpler code than the previous one in Figure A.2. Even so, there are also numerous names by default in sprites (Sprite5, Sprite6, Sprite7 or Sprite9, among others). In addition, we can find many badly initialized attributes throughout its code. The project is available at `https://scratch.mit.edu/projects/366066808/`.

Cutting Down

In this case, the code of the *Cutting Down* project almost does not have bad smells, according to the assessment of the Dr. Scratch tool. For instance, we can observe personalize names of sprites (tree1, tree2 or tree3, among others) instead of default naming in Figure A.3.

However, despite there is only one repeated structure of blocks, it is repeated in many sprites (the code of each *tree* is practically the same), causing the effect of many duplicated code. The project is available at `https://scratch.mit.edu/projects/366066161/`.

Sea Game

In Figure A.4 we can observe how in a simple code can appear different bad smells. For instance, the code of the *Sprite2* is mainly composed of repeated blocks, in addition to its default name. Moreover, we can appreciate other names by default, such as *Sprite3, Sprite4* or *Sprite5*, among others. The project is available at `https://scratch.mit.edu/projects/366067289/`.

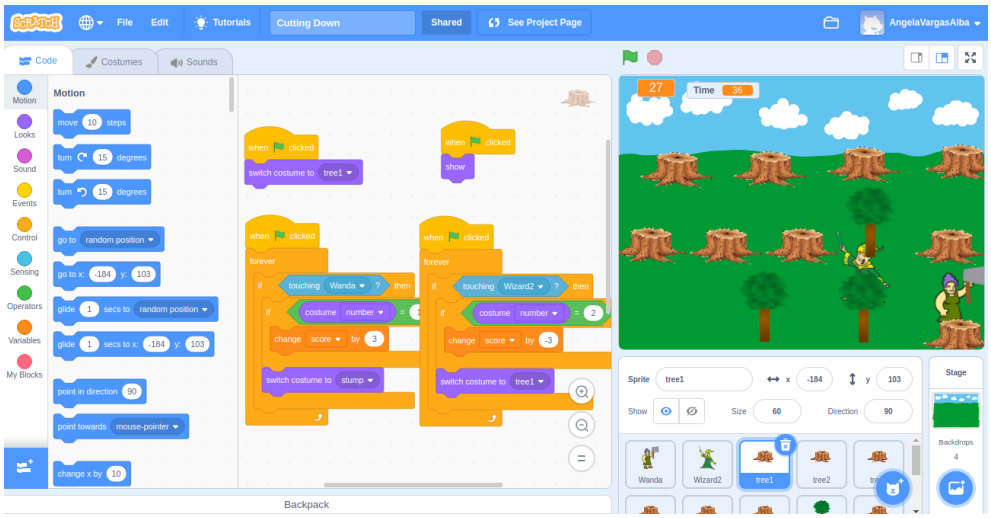


Figure (A.3) *Cutting Down* project.

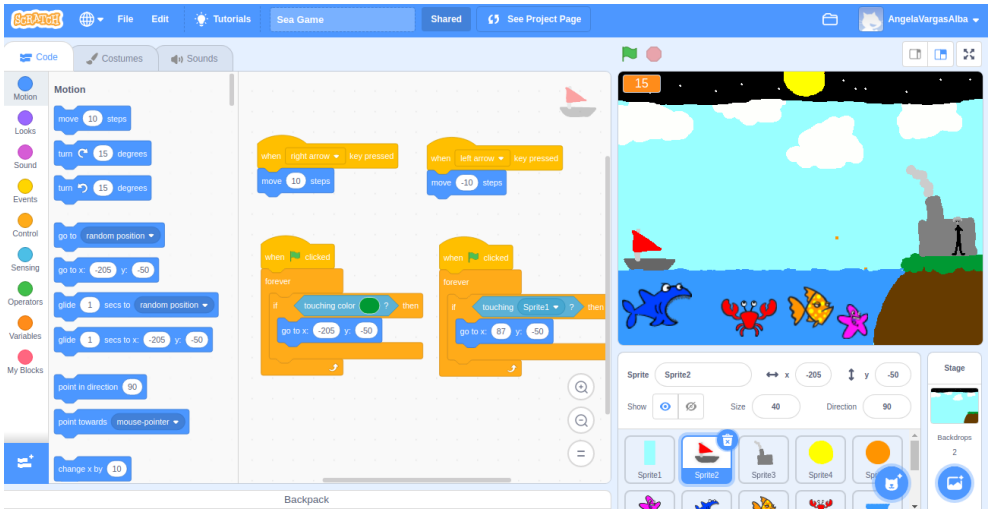


Figure (A.4) *Sea Game* project.

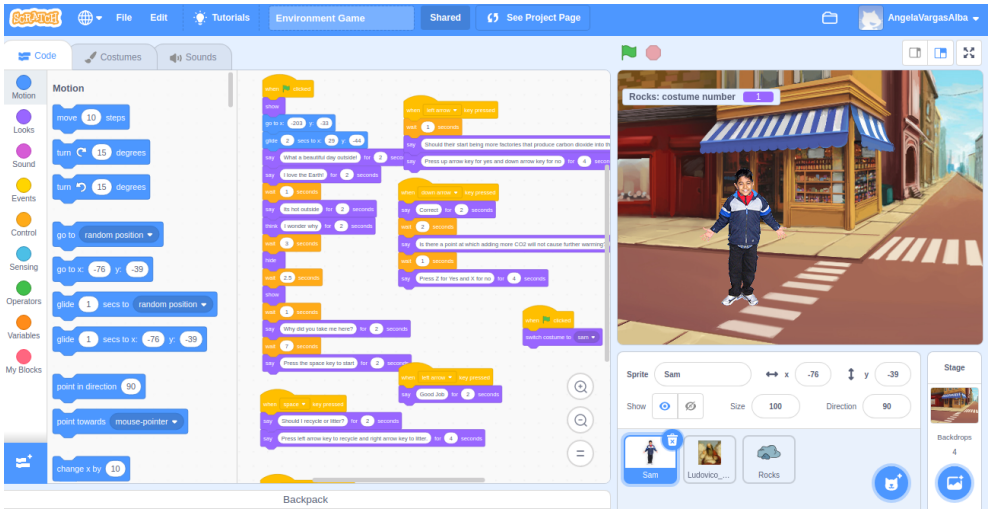


Figure (A.5) *Environment Game* project.

Environment Game

The *Environment Game* project is designed with a simple code. The program is only composed of three sprites, which are defined with personalize names, as we can observe in Figure A.5. However, the code is repetitive and the richness of blocks is scarce. The project is available at `https://scratch.mit.edu/projects/366066063/`.

Maze Game

Even though the number of bad smells in this project is almost null in the Dr. Scratch assessment, we can observe the disorder and the quality of its code in Figure A.6. We can appreciate that the majority of the scripts shares the same block structure and the same functionality. In addition, although the program does not have default naming, it mainly contains repeated sprites (Factory, Factory2, Factory3, ...). This project is available at `https://scratch.mit.edu/projects/366056709/`.

A.1 Phase 1

Regarding the Google Forms, in Figure A.7a we can observe the design of the general questions in the first phase of the experiment described in Chapter 4. This question is repeated for each project. In addition, in Figure A.7b is shown the last question related to the projects order.

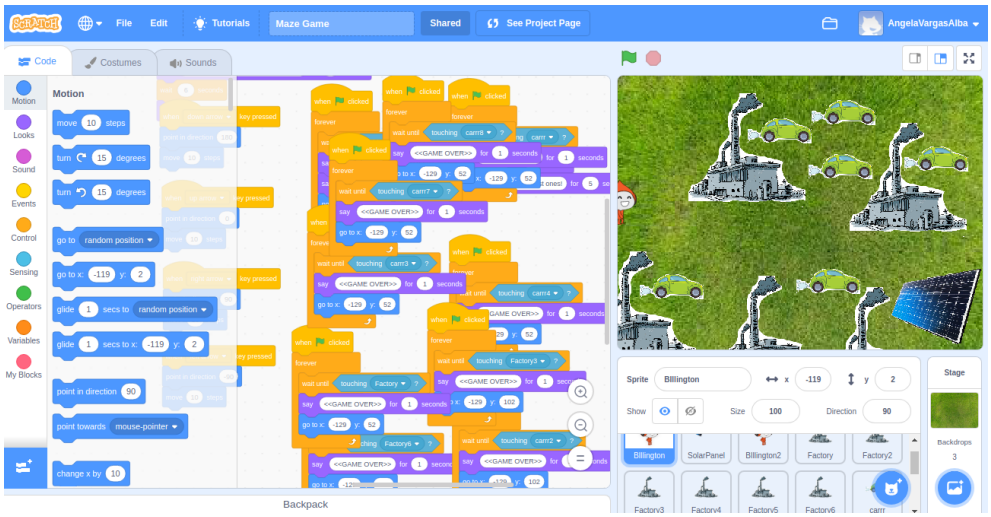
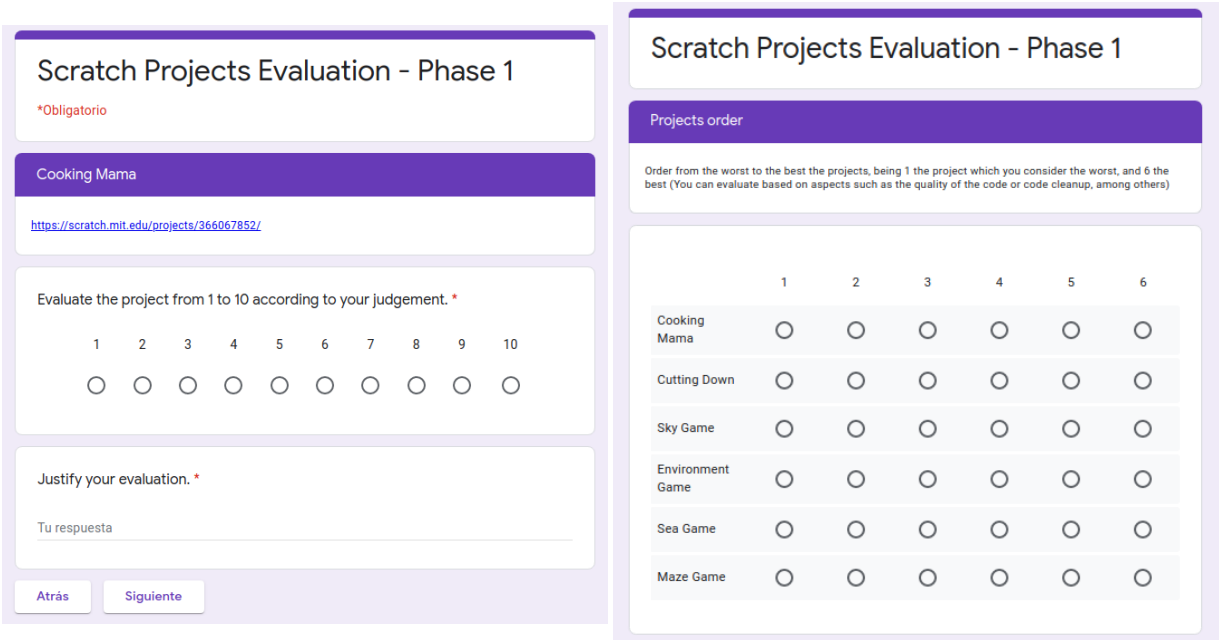


Figure (A.6) *Maze Game* project.



(a) General analysis.

(b) Projects order.

Figure (A.7) Question design in the first phase of the assessment experiment.

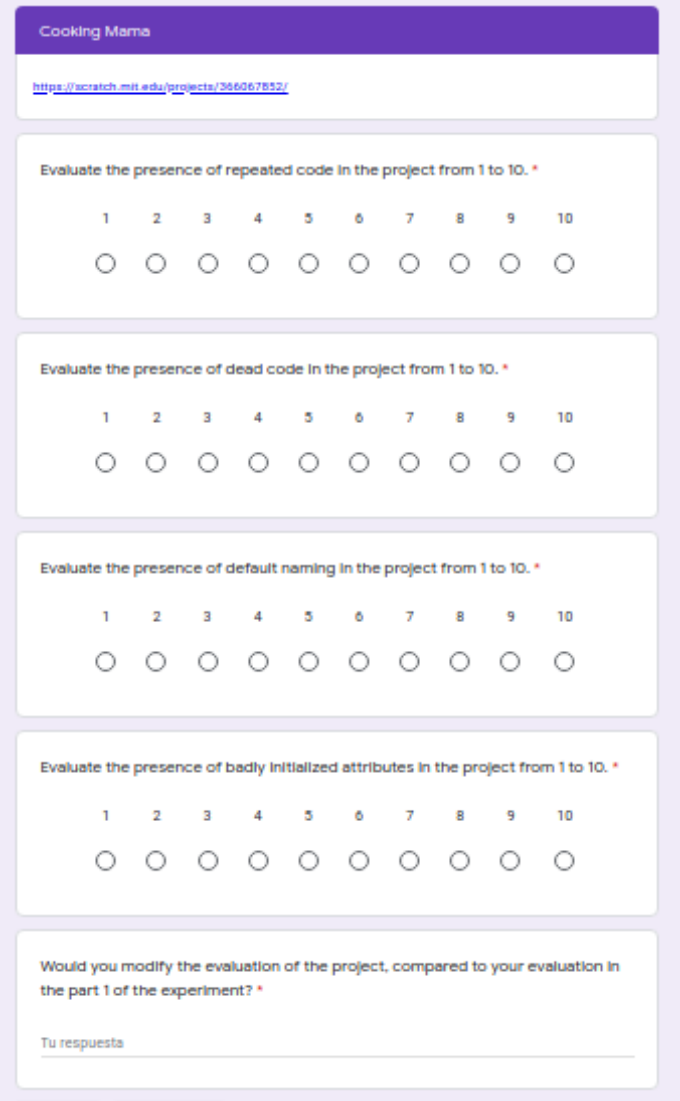


Figure (A.8) Question design in the second phase of the assessment experiment.

A.2 Phase 2

Lastly, we can observe the design of the questions in the second phase of the experiment in Figure A.8. In this case, we guide the evaluation by asking specific aspects of each bad smell. Again, this question is repeated for each project. At the end of the form, the question related to the projects order is included, as we showed in the previous form.

Bibliography

[Aivaloglou et al., 2017] Aivaloglou, E., Hermans, F., Moreno-León, J., and Robles, G. (2017). A dataset of scratch programs: scraped, shaped and scored. In *Proceedings of the 14th international conference on mining software repositories*, pages 511–514. IEEE Press.

[Calao et al., 2015] Calao, L. A., Moreno-León, J., Correa, H. E., and Robles, G. (2015). Developing mathematical thinking with scratch. In Springer, editor, *Design for Teaching and Learning in a Networked World*, pages 17–27. Springer.

[DjangoGirls, 2020] DjangoGirls (2014-2020). What is django? https://tutorial.djangogirls.org/en/django/.

[Hermans and Aivaloglou, 2016] Hermans, F. and Aivaloglou, E. (2016). Do code smells hamper novice programming? a controlled experiment on scratch programs. *2016 IEEE 24th International Conference on Program Comprehension (ICPC)*, pages 1–10.

[JavaTpoint, 2018] JavaTpoint (2011-2018). Python features. https://www.javatpoint.com/python-features.

[Kuhlman, 2009] Kuhlman, D. (2009). *A python book: Beginning python, advanced python, and python exercises*. Dave Kuhlman Lutz.

[Lutz, 2010] Lutz, M. (2010). *Programming Python: powerful object-oriented programming*. "O'Reilly Media, Inc.".

[Mangifesta and Feldfeber, 2019] Mangifesta, L. and Feldfeber, I. (2019). La importancia de enseñar programación en la escuela. https://medium.com/proyecto-mumuki/la-importancia-de-enseñar-programación-en-la-escuela-54f8dbb288fd.

[Martin, 2009] Martin, R. C. (2009). *Clean Code: A Handbook of Agile Software Craftsmanship*. Pearson Education.

[Moreno-León et al., 2015] Moreno-León, J., Robles, G., and Román-González, M. (2015). Dr. scratch: Automatic analysis of scratch projects to assess and foster computational thinking. *RED. Revista de Educación a Distancia*, 46:1–23.

[Nin, 2019] Nin, P. (2019). ¡bienvenido, scratch 3.0! https://programamos.es/bienvenido-scratch-3-0/.

[Robles et al., 2017] Robles, G., Moreno-León, J., Aivaloglou, E., and Hermans, F. (2017). Software clones in scratch projects: On the presence of copy-and-paste in computational thinking learning. In *2017 IEEE 11th International Workshop on Software Clones (IWSC)*, pages 1–7. IEEE.

[Troiano et al., 2019] Troiano, G. M., Snodgrass, S., Argımak, E., Robles, G., Smith, G., Cassidy, M., Tucker-Raymond, E., Puttick, G., and Harteveld, C. (2019). Is my game ok dr. scratch?: Exploring programming and computational thinking development via metrics in student-designed serious games for stem. In *Proceedings of the 18th ACM International Conference on Interaction Design and Children*, pages 208–219. ACM.

[Vargas-Alba et al., 2019] Vargas-Alba, A., Troiano, G. M., Chen, Q., Harteveld, C., and Robles, G. (2019). Bad smells in scratch projects: A preliminary analysis. In *Proceedings of the TACKLE 2019 workshop*.

[Wing, 2006] Wing, J. M. (2006). Computational thinking. *Communications of the ACM*, 49(3):33–35.

[Zhang et al., 2011] Zhang, M., Hall, T., and Baddoo, N. (2011). Code bad smells: A review of current knowledge. *Journal of Software Maintenance and Evolution Research and Practice*, 23(3):179–202.