Bad Smells in Scratch Projects: A Preliminary Analysis

Ángela Vargas-Alba[1], Giovanni Maria Troiano[2], Quinyu Chen[2], Casper Harteveld[2] and Gregorio Robles[1]

[1]Universidad Rey Juan Carlos, Madrid, Spain — {a.vargasa@alumnos,grex@gsyc}.urjc.es
[2]Northeastern University, Boston, MA — {g.troiano, q.chen, c.harteveld}@northeastern.edu

Abstract

Computational Thinking (CT) is an area of great relevance today. Although its skills may be developed in various ways, one of the most common tools to learn it, train it and develop it, is through programming. From software engineering, we know that problems solved through programming may have not been solved in the most appropriate way. These symptoms are known as "bad smells". This article aims to analyze the presence of several bad smells in Scratch projects and how they relate to the development of CT skills. Therefore, we make use of a dataset of several hundreds of Scratch projects with the aim of creating a game on climate change. Our results show that bad smells can be found in all types of Scratch projects, independently of the development of CT skills they require. We discuss why the learning community should address bad smells appropriately, as they may hinder the development of abstraction, reuse and other relevant skills.

1 Introduction

Bad (code) smells are symptoms that the problem to be solved is not developed in the most appropriate way. 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 [9]. Martin defines code smells as follows [3]: "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." Despite the negative effect they produce, code smells have been little investigated and analyzed in Computational Thinking (CT) research. As Hermans and Aivaloglou have found in an experiment with Scratch learners [1], we argue that bad smells hinder the proper development of CT skills in learners. Their identification should be a first step towards guiding learners towards good practices that offer them the possibility to develop themselves to their full potential.

For this reason, the main motivation of this research is to analyze to what extent bad smells are present in Scratch projects, a block-based language that is widely used around the globe to develop CT skills. Our research is similar to a previous one done on LEGO MINDSTORMS EV3 and Microsoft's Kodu [2], expanding it with information on the complexity of the projects. Therefore, we use Dr. Scratch, a tool that evaluates the richness of elements used in the programs, for evaluating the Scratch projects. Thanks to Dr. Scratch it is possible to detect different types of bad smells that are present in the code.

The remainder of this paper is structured as follows: In Section 2, we introduce and motivate the research goal and research questions that we address in this paper. A more detailed description about the definition of bad smells is summarized in Section 3. Section 4 and Section 5 describe the data set used in the study, as well as the functionality and design of Dr. Scratch in more detail. Section 6 shows the results obtained after the analysis and in Section 7 a discussion is proposed based on it. Limitations and problems found are described in Section 7.1. Finally, Section 8 contains the main conclusions and future work that we envision.

In: I. Fronza, C. Pahl (eds.): Proceedings of the 2nd Systems of Assessments for Computational Thinking Learning Workshop (TACKLE 2019), co-located with 14th European Conference on Technology Enhanced Learning (EC-TEL 2019), 17-09-2019, published at http://ceur-ws.org.

2 Research Goal and Questions

The main objective of this paper is **to analyze the presence of bad smells in a large set of Scratch projects**.

For this, the research questions that we want to address are as follows:

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 the 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 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.

3 Bad smells in Scratch

Scratch is a visual programming language formed by different blocks, designed for children and beginner programmers, which contains different bad smells related to the use of these blocks [6].

In our research, we have identified four different types of bad smells that can be present in Scratch projects: copy and pasted code (duplicate scripts) [7], 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 characteristics and impact are summarized in Table 1.

4 Research Context

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 is necessary. The data set used in this article is the same which was used for another, previous research [8]. A group of 438 students designed games for STEM using Scratch 2.0. During this process, we obtained snapshots of the process, in different periods of time, in order to show a temporary evolution[1]. The total number of projects without taking into account the replicas over time, is 711. As a result, the complete data set is comprised of 62,074 projects formed by the snapshots. With the total data set, we wanted to analyze the same 711 projects in different points of time.

All these snapshots were analyzed with Dr. Scratch, of which 2,158 were erroneous in the analysis for different reasons: the project was saved incorrectly, the code contained special characters, etc. The final set of projects analyzed was 59,916 (further details can be found in [8]).

5 Methodology

We have taken all snapshots of the Scratch projects and have analyzed them with Dr. Scratch. Dr. Scratch is a web-based tool that analyzes different categories related to computational thinking based on the blocks of the Scratch projects [4] (a screenshot of their main web page can be seen in Figure 1). It analyzes the code and, depending of the diversity of blocks used, the application gives a score to the project.

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

Table 1: Types of Bad Smells.

Bad Smell Type	Definition	Impact on Learning
Duplicate 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, costume, etc are not correctly initialized

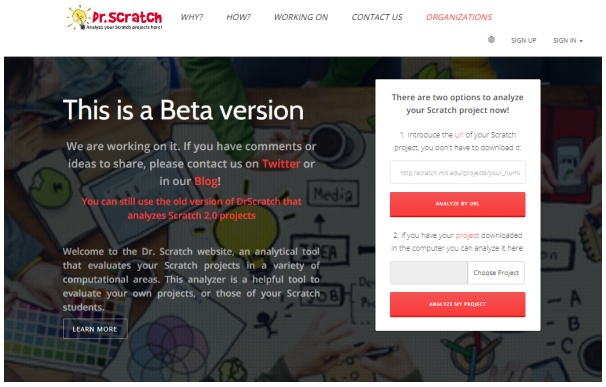

Figure 1: Main page of the web tool Dr. Scratch

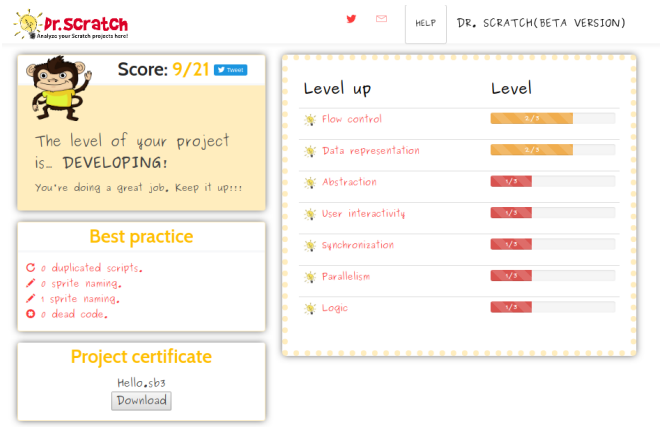

Figure 2: Example of the analysis of a Scratch project with Dr. Scratch

The outcome is a numeric punctuation based on seven categories of computational thinking: parallelism, logical thinking, flow control, user interactivity, data representation, abstraction and synchronization. For each of these abilities a project can obtain a punctuation from 0 to 3 points, according to the different blocks used in the Scratch project. In this way, the final punctuation can be from 0 to 21 total points.

Based on that, there are three different profiles of leaner: between 0 and 7 points, *Basic*, between 8 and 15, *Developing* and between 16 and 21, *Proficiency*.

Once the project is analyzed, the application will show different dashboards with these results, as it is possible to see in the Figure 2.

In addition to the former, Dr. Scratch identifies the four types of bad smells that we study in this work.

6 Results

In this Section we describe the results obtained from addressing our research questions using the previously described data set.

6.1 To what extent are bad habits present?

As shown in Table 2, bad smells can be found in almost all projects in our data set – over 97% of the projects have at least one bad smell.

Table 2: General presence of Bad Smells (RQ1)

Projects with bad smells	Projects without bad smells
58,162	1,754
97,07%	2,93%

We expected a high share of projects having bad smells, but this result is a surprise for us, as the presence of bad smells is not only more frequent than expected, but almost general.

6.2 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 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.

In Figure 3 we can observe the distribution of each set of projects. We can observe that more than 50% of those projects with not 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 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.

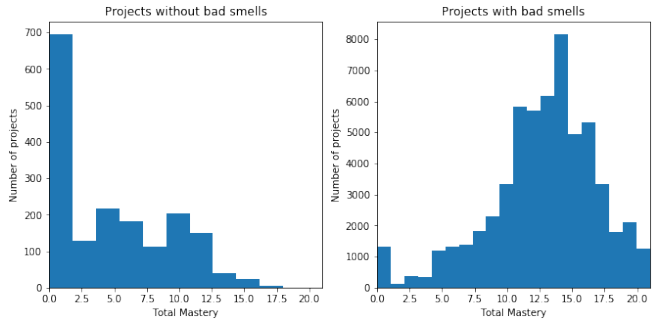

Figure 3: Distribution for projects without bad smells and with bad smells (RQ2).

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.

6.3 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 4 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. Both curves have been normalized to their maximum values. We can observe how the two curves run almost in parallel (up to 8 points they share the same ratio, and then they share a ratio of around 0.5 up to 21 points, where the ratio is over 1).

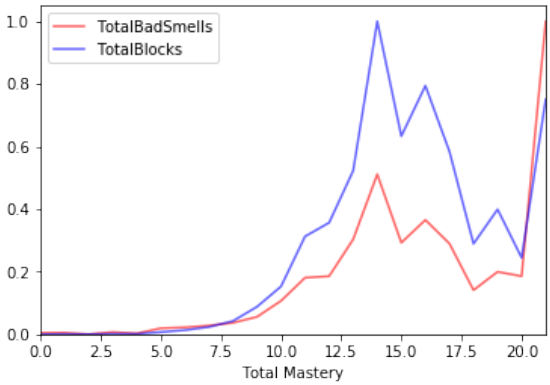

Figure 4: Relation between the total number of bad smells and the total number of blocks for each mastery level (RQ3).

Table 3 offers more details about this situation.

6.4 Can we find a relation among specific bad smells?

From Figure 5 (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, and 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 (Sprite 1, Sprite 2, etc) instead of personalizing them.

6.5 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 6, we have represented the percentage of projects that have at least a specific type of bad smell for each level of CT proficiency. The percentage of projects from users with the basic level that have bad smells is much smaller than the percentage of projects that require a proficiency level. This result indicates that all bad smells have an incremental evolution with the increase of CT efficiency. So, it seems that bad smells appear in early phases and instead of disappearing with the

Table 3: Detailed information on number of blocks and bad smells for each mastery level (RQ3).

Total Mastery	Total Projects	Total Blocks	Mean Blocks	Median Blocks	Total Bad Smells	Mean Bad Smells	Median Bad Smells	Total Blocks / Bad Smells
0	1747	198	0.11	0	2087	1.19	1	0.09
1	279	2096	7.51	1	2200	7.89	1	0.95
2	171	702	4.11	3	468	2.74	1	1.50
3	464	3519	7.58	4	2914	6.28	0	1.21
4	425	2902	6.83	5	1414	3.33	0	2.05
5	1325	15365	11.60	9	7268	5.49	1	2.11
6	1417	28878	20.38	16	8298	5.86	2	3.48
7	1470	51024	34.71	22	10304	7.01	2	4.95
8	1935	92437	47.77	29	13459	6.96	2	6.87
9	2349	191908	81.70	55	20221	8.61	3	9.49
10	3473	334075	96.19	66	38376	11.05	3	8.71
11	5900	683490	115.85	82	64702	10.97	4	10.56
12	5786	779130	134.66	103	66401	11.48	4	11.73
13	6206	1142705	184.13	142	108307	17.45	7	10.55
14	8173	2184999	267.34	201	182128	22.28	10	12.00
15	4954	1383152	279.20	193	104416	21.08	7	13.25
16	5346	1735651	324.66	255	130280	24.37	10	13.22
17	3352	1276143	380.71	291	103394	30.85	16	12.34
18	1795	631899	352.03	279	50581	28.18	10	12.49
19	2092	872027	416.84	342	71391	34.13	14	12.21
20	960	533958	556.21	507	66419	69.19	26	8.04
21	296	1639757	5539.72	7183	355809	1202.06	1505	4.61

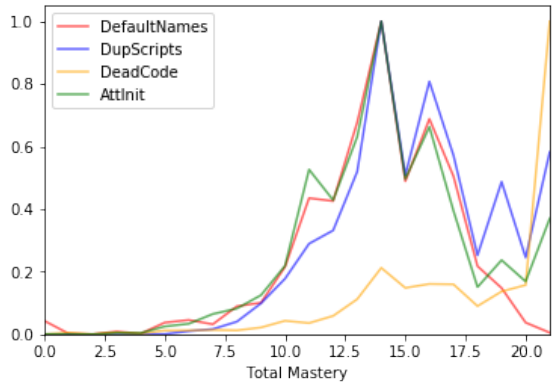

Figure 5: Evolution of the different types of bad smells with CT mastery (RQ4).

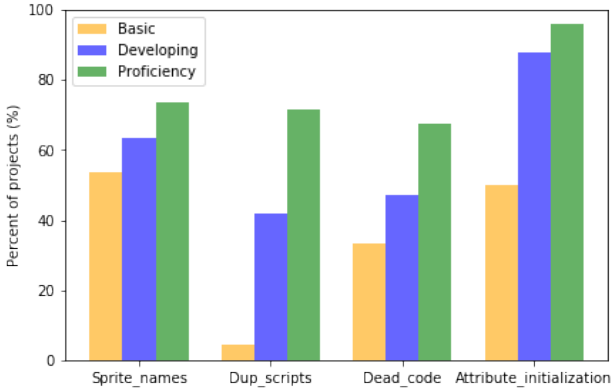

Figure 6: Presence of bad smells for each proficiency level (RQ5).

development of more advanced CT skills, they become more prominent.

7 Discussion

We have observed that bad smells are very common in Scratch projects. The results indicate as well that bad smells do not limit the development of a project, because it is possible to create projects at the proficiency level even with a large amount of bad smells.

We argue that researchers, educators and learners should devote more effort in avoiding the presence of bad smells in learning projects. The very nature of bad smells makes them difficult to identify. They are not errors and a program could run perfectly having many of them. However, their effects are very well known in Software Engineering research. These effects are in the long term, when maintainability of a software system is considered. In such a situation, the software has to be understood and changed, and in that situation is where these bad smells become more prominent. The

large presence of bad smells indicates that understanding and changing code is not among the priorities of the projects under study, although "good code" has always been considered as that one that is easy to understand and to change. That is why we think that with the current presence of bad smells learners do not achieve their full potential of CT development skills. In our opinion, further research and tools should be envisioned and created to address this issue.

7.1 Limitations

As any research, our work comes with a number of limitations that can be seen as threats to its validity.

The first one is related to our methodology, and in particular to the limited set of bad smells that we can identify. We are sure that many other types of bad smells could be thought of in Scratch. On the other hand, we use as complexity metrics the CT scores provided by Dr. Scratch. While there have been some research that has studied how Dr. Scratch correlates with other complexity metrics [5], this is always a correlation and not causation.

We have studied projects from a specific environment, which may not be representative for all Scratch projects, so the generalization of results is a threat to validity. However, it should be noted that this is a case study, with the aim to raise attention on this matter. We should analyze a wider data set which includes different sectors of programming with Scratch, such as stories, music, animations, among others. We do not know if the behavior of bad smells could be different in other areas of programming.

8 Conclusions

Bad smells are common in Scratch projects, from basic to proficient. As the complexity increases, bad smells do not disappear, so learners can create projects that demand a high development of CT skills having bad smells in them.

We ask for further research on this topic.

Acknowledgments

This work has been co-funded by the Madrid Regional Government, through the project e-Madrid-CM (P2018/TCS-4307). The e-Madrid-CM project is also co-financed by the Structural Funds (FSE and FEDER).

References

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

[2] F. Hermans, K. T. Stolee, and D. Hoepelman. Smells in block-based programming languages. In *2016 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, pages 68–72. IEEE, 2016.

[3] R. C. Martin. *Clean code: a handbook of agile software craftsmanship*. Pearson Education, 2009.

[4] J. Moreno-León, G. Robles, et al. Analyze your scratch projects with dr. scratch and assess your computational thinking skills. In *Scratch conference*, pages 12–15, 2015.

[5] J. Moreno-León, G. Robles, and M. Román-González. Comparing computational thinking development assessment scores with software complexity metrics. In *2016 IEEE global engineering education conference (EDUCON)*, pages 1040–1045. IEEE, 2016.

[6] M. Resnick, J. Maloney, A. Monroy-Hernández, N. Rusk, E. Eastmond, K. Brennan, A. Millner, E. Rosenbaum, J. S. Silver, B. Silverman, et al. Scratch: Programming for all. *Commun. Acm*, 52(11):60–67, 2009.

[7] G. Robles, J. Moreno-León, E. Aivaloglou, and F. Hermans. 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, 2017.

[8] G. M. Troiano, S. Snodgrass, E. Argımak, G. Robles, G. Smith, M. Cassidy, E. Tucker-Raymond, G. Puttick, and C. Harteveld. 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, 2019.

[9] M. Zhang, T. Hall, and N. Baddoo. Code bad smells: a review of current knowledge. *Journal of Software Maintenance and Evolution: research and practice*, 23(3):179–202, 2011.