Measurement and Visualization of Programming Processes of Primary School Students in Scratch

Alexandra Funke
Technical University of Munich
TUM School of Education
alexandra.funke@tum.de

Katharina Geldreich
Technical University of Munich
TUM School of Education
katharina.geldreich@tum.de

ABSTRACT

Currently in many countries efforts are undertaken to bring programming education into the early levels of childhood education, like primary school or even kindergarten. Therefore, it is becoming more and more important to gain insight into which teaching methods and content would be appropriate for young students of primary levels. Besides the analysis of the results of such courses, it is particularly interesting, in which way the programming processes of the children take place and if there are distinguishable types of young programming learners. To illustrate the processes and to explore differences and special features of the individual approaches we developed a new visualization technique for the example of the programming language Scratch.

CCS CONCEPTS

• **Social and professional topics → Computer science education**; **Children**;

KEYWORDS

Computer science education; visualization; scratch; programming

ACM Reference Format:
Alexandra Funke and Katharina Geldreich. 2017. Measurement and Visualization of Programming Processes of Primary School Students in Scratch. In *Proceedings of WiPSCE '17*. ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/3137065.3137086

1 INTRODUCTION

Computer Science (CS) has to overcome several challenges. Schools and universities are confronted with different misconceptions and prejudices towards CS [1] which manifest themselves from an early age [5]. In order to prevent students from developing negative attitudes, one approach is to introduce Computer Science concepts like programming as early as in primary school to provide opportunities for children to experience technology and CS. At the same time, the discussion about the necessity of computer science and programming in childhood education is growing steadily [7].

To find out which teaching methods and content would be appropriate for German primary schools, we designed an introductory programming course for fourth graders. Within this, we used, as many other researchers, Scratch as a programming environment.

Although a predominant goal of such courses is to change the students' attitudes towards CS, it is also important to gain insight into how they create programs and how they apply the learned programming concepts. To figure this out, we developed a visualization technique to visualize the results of Scratch programming course in order to better understand the process. Our research question in this work was: *How can programming processes be visualized so that differences are observable?*

WiPSCE '17, November 8–10, 2017, Nijmegen, Netherlands

ACM ISBN 978-1-4503-5428-8/17/11…$15.00
https://doi.org/10.1145/3137065.3137086

2 BACKGROUND

As we have also experienced, the development of a perfect visualization is often a big challenge [3]. There are different classifications of visualization. In [3] the authors named three categories of visualization: a) Scientific visualization to understand physical phenomena or mathematical models; b) Software visualization, which helps people to learn the use of e.g. software; c) Information visualization which visualizes information with the use of spatial or graphical representations. A more detailed classification can be found in the periodic table of visualization [4]. For the six main categories *Data Visualization*, *Information Visualization*, *Concept Visualization*, *Strategy Visualization*, *Metaphor Visualization* and *Compound Visualization*, the authors described suitable methods. Further, they divided the methods in *Process Visualization* and *Structure Visualization*. This work focuses on *Process Visualization* methods of the *Information Visualization* class. Examples for this in the periodic system are Cycle Diagram, Petri Net, System Dynamics / Simulation, Timeline, Flow Chart and Data Flow Diagram.

Very popular diagrams in computer science are behavior and structure diagrams of the Unified Modeling Language (UML)[6]. To visualize processes and workflows in software, there are, for example, Activity Diagrams, Communication Diagrams, and Sequence Diagrams. For this work, the most interesting UML diagram type is the Sequence Diagram. It shows object interactions during a program execution arranged in a time sequence [6].

2.1 Visualization of programming processes in Scratch

As shown above, there are currently many visualizations for software and hierarchical data. The problem with most diagrams and images is that they show parallel or hierarchical data. In contrast, a programming process is sequential; there are no hierarchical structures or alternatively branches. But still, it is important to see different sprites in the same diagram. For this purpose, we found no appropriate visualization.

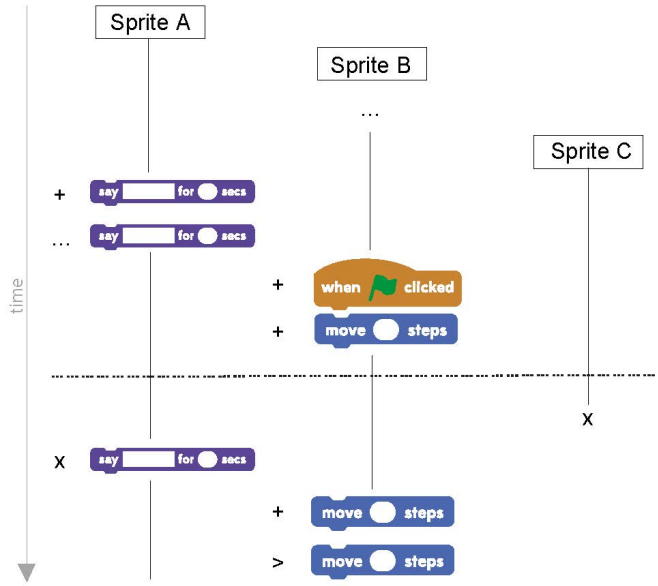

Figure 1: Example of a *Development Diagram*

Even the described UML diagrams which are common in software engineering are not usable in this case. While it is possible to model interactions and communication within a program or between program and user, this is not the use case in this work.

We developed two different diagram types to show the results of each student in a clear and comprehensible way.

2.2 Development diagram

We called the first created diagram type *Development Diagram*. It is based on UML sequence diagrams which show how objects operate with one another and in what order. Instead of this original diagram, *Development Diagrams* describe the programming process for all Sprites of one Scratch project, but no communication between them. Fig. 1 shows an example *Development Diagram*. It starts with the first Sprite which is added to the project. Because the time-line runs from top to bottom, every new step in the programming process is a new line in the diagram. After adding Sprite A, the user added Sprite B. The editing of Sprite B is shown with the three points in line 3. To describe if a block was added, edited, deleted or moved to another position within the script, the image of the block is displayed with the associated icon in front of it (e.g. a plus icon for adding an element). A dashed horizontal line stands for a run of the script. In this example, the project is started with a click on the green flag. Sprite C is deleted without editing the script or the sprite itself.

2.3 Sprite diagram

The second diagram type we called *Sprite Diagram*. It is in the broadest sense a mix between an object diagram and an activity diagram. *Sprite Diagrams* describe the programming process for one Sprite of the Scratch project. Fig. 2 shows an example *Sprite Diagram*. It starts with the name of the Sprite.

Because the time-line runs from top to bottom, every new step in the programming process is a new line in the diagram. The columns

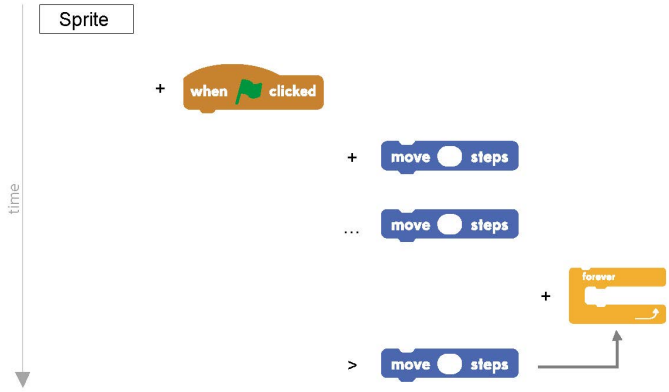

Figure 2: Example of a *Sprite Diagram*

of the diagram are blocks which were used in the project. Because it is interesting, which exact block was added or deleted, the changes of the same block are written among each other. Some block types, like *forever* or *if_then* can contain other blocks. To visualize which blocks were added within *forever* or *if_then*, arrows are used.

3 CONCLUSION AND FUTURE WORK

With the developed diagram types, it is easily possible to find important points in the programming processes of the students. To show this, we will code and analyze data from our programming courses in a future work. Based on this and the created visualizations, we will develop a system for the automated generation of the visualizations from a list of codes. Afterwards, the visualizations will be automatically generated and analyzed. One goal of our work is to find out if we can distinguish different programming types between the students, the classes or specific groups of children. We think that the visualizations together with screen captures and video recordings will play an important role in this step. In a previous work of ours [2] we found that the children created three project types in particular: Story, Game, and Animation. We wonder if the project type relates to the programming process and the other way around.

REFERENCES

[1] A. Funke, M. Berges, and P. Hubwieser. Different Perceptions of Computer Science. In *2016 International Conference on Learning and Teaching in Computing and Engineering (LaTICE)*, pages 14–18. IEEE, 2016.

[2] A. Funke, K. Geldreich, and P. Hubwieser. Analysis of Scratch Projects of an Introductory Programming Course for Primary School Students. In *Proceedings of the 2017 IEEE Global Engineering Education Conference (EDUCON)*, pages 1233 – 1240. IEEE, 2017.

[3] M. Khan and S. S. Khan. Article: Data and information visualization methods, and interactive mechanisms: A survey. *International Journal of Computer Applications*, 34(1):1–14, November 2011.

[4] R. Lengler and M. J. Eppler. Towards a periodic table of visualization methods of management. In *Proceedings of the IASTED International Conference on Graphics and Visualization in Engineering*, pages 83–88, Anaheim, USA, 2007. ACTA Press.

[5] K. Prottsman. Computer science for the elementary classroom. *ACM Inroads*, 5(4):60–63, 2014.

[6] J. Rumbaugh, I. Jacobson, and G. Booch. *Unified Modeling Language Reference Manual*. Pearson Higher Education, 2004.

[7] H. Topi. Gender imbalance in computing. *ACM Inroads*, 6(4):22–23, 2015.