Clickstream for learning analytics to assess students' behavior with Scratch

Daniel Amo Filvà
La Salle, Universitat Ramón Llull
Departament d'informàtica
Spain
daniel.amo@salle.url.edu

Marc Alier Forment
Universitat Politècnica de Catalunya
UPC Campus Nord, Ed. Omega, Jordi Girona 1-3
Spain
malier@essi.upc.edu

Francisco José García-Peñalvo
Universidad de Salamanca
Departamento de Informática y Automática
Spain
fgarcia@usal.es

David Fonseca Escudero
La Salle, Universitat Ramón Llull
Departament d'arquitectura
Spain
david.fonseca@salle.url.edu

María José Casañ
Universitat Politècnica de Catalunya
Spain
mjcasany@essi.upc.edu

Abstract

The construction of knowledge through computational practice requires to teachers a substantial amount of time and effort to evaluate programming skills, to understand and to glimpse the evolution of the students and finally to state a quantitative judgment in learning assessment. The field of learning analytics has been a common practice in research since last years due to their great possibilities in terms of learning improvement. Both, Big and Small data techniques support the analysis cycle of learning analytics and risk of students' failure prediction. Such possibilities can be a strong positive contribution to the field of computational practice such as programming. Our main objective was to help teachers in their assessments through to make those possibilities effective. Thus, we have developed a functional solution to categorize and understand students' behavior in programming activities based in Scratch. Through collection and analysis of data generated by students' clicks in Scratch, we proceed to execute both exploratory and predictive analytics to detect patterns in students' behavior when developing solutions for assignments. We concluded that resultant taxonomy could help teachers to better support their students by giving real-time quality feedback and act before students deliver incorrectly or at least incomplete tasks.

Keywords: learning analytics; clickstream; scratch; programming; big data

1 Introduction

Learning Analytics (LA) and Educational Data Mining (EDM) are both analytical approaches to enhance and optimize learning environments [36]. However, there are some key distinctions between them. EDM is a branch of data mining and machine learning [8, 26, 34] to analyze educational data, where its origin is

in educational software and student modeling. Hence, this approach is more focused on computer and automation. On the other hand, the discipline of LA is more human-centered. It allows to analyze student behavior patterns when they interact with tools and online learning environments, set them in context with their learning outcomes and draw conclusions to enhance the evaluation or improve the learning process through human judgment [39].

Programming environments can be adapted and integrated into a Learning Analytics system. Therefore, any of the related steps of a programming task can be analyzed, namely: first actions of the student after the presentation of the activity, analysis, and design of the solution, the process of development, and delivered code.

The results obtained by this type of analysis can be considered as very valuable information for any teacher since it will allow them to know how each student is evolving, what their status is in relation to the proposed tasks and what is their possible risk to suspend according to trends extracted from previous analyzes. Therefore, the student's follow-up in a programming environment could help teachers to provide better support, enhance tutoring and adapt content or activities.

In a programming activity, each student has a different style and therefore can develop a unique and different solution in comparison to the rest of their classmates [29]. This means that the teacher must spend a great deal of time analyzing each of the possible solutions delivered [5, 16]. To know what the student has done or what their behavior has been during the development of the proposed activities, we can benefit from automatic data collection and analytical techniques. The application of Learning Analytics in programming practice now makes much more sense [3, 23].

From the organizations' point of view, the introduction of data analytics can be seen as a deep and accelerating transformation with regard to processes, activities, competencies, and models, in order to take advantage of the changes and opportunities offered by the inclusion of digital technologies into an organization. However, this advantage is only possible if the information systems of the organizations are aligned with these new technologies. Therefore, in this work, we propose the use of the clickstream technique for the collection and analysis of students' interaction data with big data techniques to understand and improve their learning process [26, 27, 30, 36, 39].

Clickstream is defined as the sequence of actions taken by visitors through a website. Clickstream analytics is a technique used in web applications, typically in e-commerce, to know how visitors behave in a website. Through the collection of clicks in the different parts of the web pages and an analytical and visualization tool [6, 22] you can get an idea of the behavior of the visitor. This information can be very valuable in refining the user experience and adapting business strategies to maximize the conversion of visits into sales [9, 28].

We propose that teachers use this technique of data analysis in visual programming environments like Scratch. This will allow knowing how students proceed in these learning and programming environments and thus improve their tutoring and support to students [10, 37]. The evaluation can also be improved due to teachers will obtain objective information about students' deliveries.

We also propose a modification of the Scratch tool to incorporate the click collection technique. Students can develop applications in Scratch through the stacking of blocks of instructions and in a very visual way. This also facilitates the learning of programming concepts. The modifications carried out allow capturing all the clicks made in this visual programming environment.

The clicks allow reconstructing in some way everything that the student has done during the development of the solution. This allows knowing interesting aspects such as if he has worked in class, which blocks have used, if he has applied the learned concepts or if he has used the structures worked in the classroom. With this information, we can identify and classify the different ways of approaching the assigned programming task. The collection of clicks [38] can discover deeper aspects of students' behavior such as different programming styles [5] according to the clicks done in zones of high concentration of clicks.

This project is born from the opportunity to conduct workshops with students and teachers willing to develop programming activities in Scratch. With this outstanding scenario, we have developed a technology able to capture, store and analyze students' interactions in this programming environment. It is expected that the analysis will yield results related to students programming style in this type of visual environment. It is also expected to identify possible behavior patterns that could help teachers to improve evaluation, tutoring, and follow-up of the students. These last two affirmative sentences are based on the work of the authors Vihavainen et al. [38] in which they use the keystrokes and clicks to find patterns of behavior in novice students' programming. His work, which analyzes conventional programming environments based

on clicks, created expectations to us regarding the recognition of behavior patterns in visual programming environments such as Scratch, since there are conceptual similarities between both environments. In the Material and Methods section, we specify the architecture design and analytical models used.

It is intended to achieve the following objectives with the use of the developed tool:

- Create new classifications of ways to learn to program and programming styles.
- Capture the inputs of this classification.
- Help teachers to teach, tutor and evaluate better.

The paper is structured into 4 more sections. Section 2 introduces the theoretical background where Scratch and clickstream are explained. Section 3 deepens into the technologies and computer science architectures that could extract new and expected results about students' behaviors. Section 4 exposes the results where both quantitative analysis and visualizations are explained. Section 5 is about the conclusions and closes the paper where actual limitations and future works are exposed.

2 Theoretical background

In this section we present the research context and also the Clickstream and Scratch as the method and technology to extend the research context to practice. In the research context, we argue the need to tackle a different approximation to the analysis of students programming. In the Clickstream section, we present in detail the method used to collect students interactions while learning programming. Finally, we introduce Scratch as the technology that helps students to learn to program and develop their programs through clicks.

2.1 Research context

Since 2010, the analysis of learning has become relevant in the field of research to improve learning and the educational context in general. There is a concern in the scientific community to understand how students learn to program from their beginnings to advanced levels [16–18, 24]. There is no single definition although the one proposed by Erik Duval [21] that seems to be the most accurate to explain this situation when he defines "Learning Analytics is about collecting traces that learners leave behind and using those traces to improve learning".

Different proposals encourage the use of learning analytics to improve the teaching of programming. Sherman and Martin [35] propose an analytical approach to extract patterns of behavior in student developers of App Inventor projects, while other authors try to glimpse the progression of computational

skills in such programming platform [12, 41]. In comparison with our work, which also aims to identify patterns of behavior, our approach is based on clicks instead of snapshots of the source code. This is an approach not used so far in this type of programming environment that we hope can provide new and interesting insights to the scientific-educational community.

Other authors [38] have proposed to understand how novice programmers approach their first lines of code in conventional programming environments. Their approach to data collection is mixed, based on the keystrokes and events in the IDE, including clicks. However, these authors have focused on the keystrokes and have not gone deeply into the analysis of clicks.

In business, it is very common to analyze processes, resources, and tasks. In e-commerce, it is even an indispensable requirement. As a result, different techniques have been developed to analyze the behavior of customers. In this paper, we propose the use of the clickstream technique used in business. The use of this technique is applied above all in the e-commerce web pages [33]. In this way, the behavior of the clients can explain their way of acting and create a taxonomy of behaviors to offer them more related products or to guide them better through the website. In education, this technique can be applied to better understand students and to personalize learning. We understand that clicks will offer additional information and complement current research.

Blikstein et al. [2] also used EDM techniques and learning analytics in student code snapshots during their programming tasks. Blikstein intends to use these approaches to automate assessments that would otherwise be impossible for teachers to detect or very expensive in time. Our objective and methods of analysis resemble this from the author, but the data capture differs and the modeling of capture, prediction, analysis, and visualization.

2.2 Clickstream

The computation and analysis of the flow of clicks on a website is an important factor to improve the decisions of businesses that have an appearance in web environments [7, 25]. Different stochastic, statistics and mathematics theories conform models that help companies decide which strategies they could take on different types of websites. This is the case of Markov chains, a stochastic model that describes a sequence of possible events in which the probability of each event depends solely on the state reached in the previous event. The problem with this model is that it has no memory and does not take into account the demographic

or sociological factors [25]. Consequently, different authors have proposed new models or modifications to previous models to improve accuracy. Montgomery et al. [25] point out that using memory in the paths detected in the clickstream increases up to 40% the accuracy of the prediction of actions in contrast to a 7% obtained with models that do not have memory. Other authors [14] point out that it is possible to use learning algorithms in clickstream data. However, [14] points out that research in this context is in a very first stage and that contributions are expected to improve in neural networks, genetic algorithms or supervised machine learning algorithms such as support vector machines. In short, these algorithms help analyze information captured by websites.

D. Wilson [9] states that a company can learn from the information captured from its visitors through their clicks such as what they are looking for or different parts they are spending time. The combination of this information with other visitor data such as geographic or demographic data extends their learning. Consequently, the results of the analysis collected from the clickstream through analytical algorithms can be used to design, adapt and evaluate web pages as well as the making of marketing programs. Wilson [40] illustrates how the conjunction between clickstream technique and network analysis is a productive approach to learn from the navigations' paths of new and recurring visitors in websites to determine how those visitors behave in front of marketing offers and even predict navigation behaviors.

In the educational context it is possible to transfer the knowledge acquired and the techniques used in business clickstream research [1, 13, 20, 32, 39]. In addition, the results can help to understand how new programmers learn to program in visual programming environments [11]. Eguiluz et al. [11] set a precedent for the use of clickstream analytics in a visual programming environment based on blocks. His research and ours have gone by the hand in hand in time, although in two well-differentiated ways. Eguiluz et al. [11] apply a more traditional clickstream and closer to the initial definition where clicks are made on programming blocks instead of web links. In this sense, they use the clickstream to analyze the blocks that students have used, which means that they analyze the code resulting from clicking on several blocks of programming. They analyze the time needed to solve a challenge (in milliseconds), the length of the code (number of blocks), the depth of the code (number of nests in the control structures), the number of code changes and the number of attempts to each participant and challenge.

Our proposal offers a new meaning for the clickstream technique that embraces additional information from the student. We can analyze the code resulting from clicking through the different blocks, but beyond this, we intend to know what the students actually have done in the block-like visual programming environment, in what places they have clicked, when they have tried to drag a block, in which zones they have more clicks, in which aspects of the program they have entertained or concentrated more, or among other things, which path they draw between the use of one block or another. We intend to define a deeper clickstream that provides a global vision of student behavior in this type of visual programming environment.

2.3 Scratch

Scratch is a visual programming language where users can create online projects and develop them into almost anything by using a simple block-like interface. The MIT Media Lab's Lifelong Kindergarten group, led by Mitchel Resnick, developed the first desktop-only version of Scratch in 2003 with the purpose to aid young people to learn programming. This platform has been our field of tests and research to extract results and conclusions in the study of behaviors in visual programming environments. The reason for using this tool is that its use is widespread in schools and institutes around the world [31] and even outside educational environments so that students learn to program and develop technological solutions.

The union of Scratch and Clickstream has resulted in a new field of research that opens new approaches to the study of students' behavior in visual programming environments. The next step we are already taking is to extrapolate the results of the research to non-visual environments at the university and post-university levels.

3 Material and Methods

In the pre-development phase of the tool, we reflect on the need to apply it on closed or open results-oriented learning tasks. Different authors who have analyzed these two types of results decided to first analyze closed results tasks [5, 38] due a specific solution is expected so is easier to conduct a research in a qualitative manner because of the cost of recording, transcribing and analysis of the data. The studies of such authors are an evolutionist, since their objective is to discover and create a framework for the analysis of interactions in programming environments. Therefore, they start from an empty metadata framework of analysis of student behavior in their learning of programming methodologies.

Our scenario is completely different. One of our proposed objectives was to compare our results with those of other authors to validate them and incorporate them into the data offered to teachers. We hope to start from a previous frame with which to compare our results regarding the collection technique based on clicks [5]. However, we understood that the analysis of clicks could provide additional classifications to offer teachers richer and more efficient data analytics to enhance as evaluators and tutors. Consequently, we freely take the path of using open results tasks and let the students freely develop their solutions.

To capture the clicks, store them, analyze them and draw conclusions, we have developed a solution supported by an architectural model based on events, a web service to send and store the interactions and a predictive model to forecast understandable results for teachers. The resulting visualizations and insights could be incorporated into the Students Progress Snapshot (SPS, from now) [1] so that the teacher has full access to information related to the learning of their students.

3.1 Previous Work

The present work is the continuation of our previous one in the application of learning in virtual learning environments [1] such as Moodle. Moodle captures the interactions of students through web access to tasks, activities, documents, profiles and other resources made available by teachers. In fact, Moodle is carrying out a data collection through clickstream that can be used to perform an exploratory and predictive analysis to visually show teachers the past, present and future status of their students. In the previous work, we made a dashboard for the teacher based on the logs of Moodle to visualize the status of the student in terms of access and time dedication on tasks and resources and the quarterly trends of their behavior.

This research aims to expand the context of the previous work by offering teachers more details about the evolution and behavior of students in visual programming environments. If the previous SPS offered general information about the student, the updated SPS, considering the current additions, allows the teacher to have a more detailed, specific and predictive view of their students in relation to the programming tasks. Consequently, the results of the present investigation are considered as a detailed extension of the SPS.

SPS is a useful approach for teachers that gives them the opportunity to see the interactions of their students beyond the basic Moodle's reports. It has been proved that this visual profile approximation with embedded textual information helps teachers to make right decisions and help students in their learning. Our aim is to

continually improve this tool to offer more insights and actionable information to teachers. Moodle does not offer a unique report. This hinder teachers to be agile in the search of students interactions and finally to make good decisions. The reason to evolve and to focus on the development of our tool is to unify all the students interactions in one place to give the possibility to teachers to access to that information as fast as possible, in real time and with easy understandable visualizations.

The following figures show a summary of a visual profile of a student already developed in the SPS. This panel of visualizations is extended with figures 9, 10, 11, 12 and 13 resulting from the current investigation and presented in the results section. Figure 1 shows the summarized visual profile.

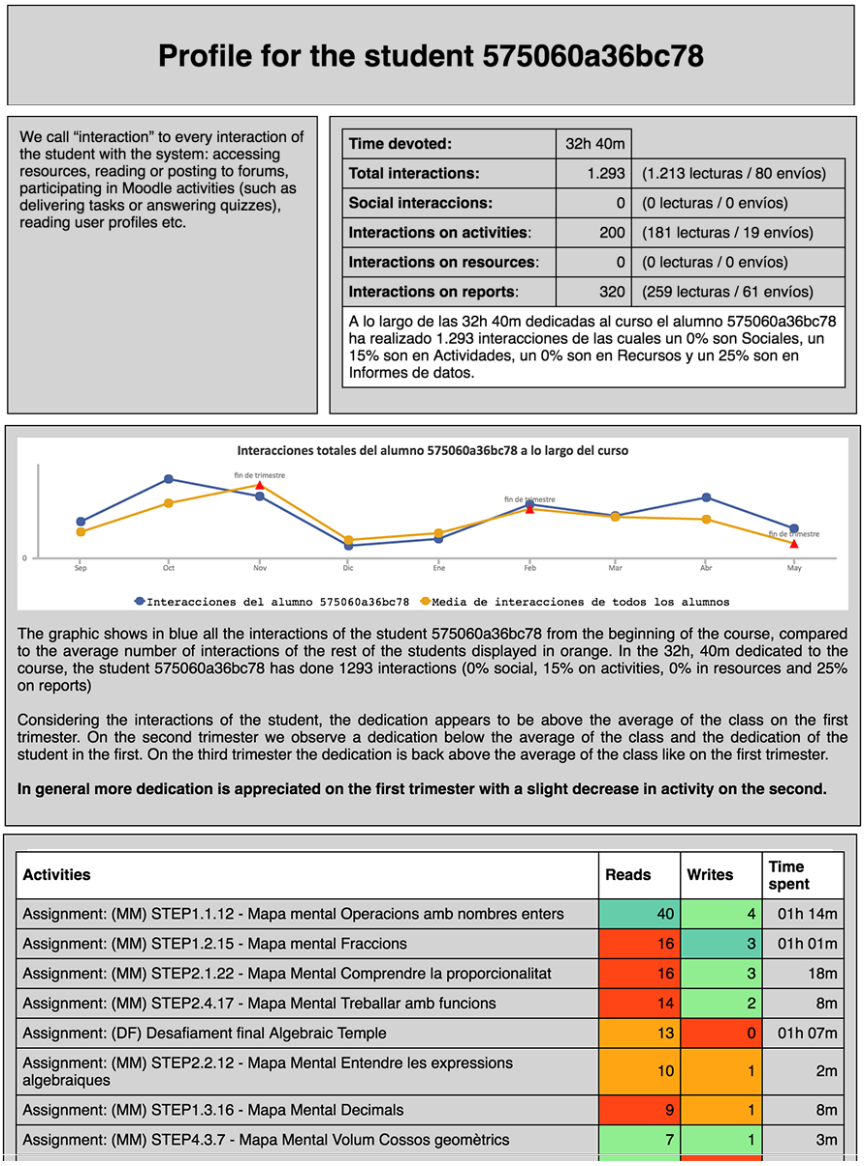

Figure 1. Summary of student's interactions.

3.2 Click collection

To diminish development costs, we have decided to add new layers to existing solutions instead of developing new ones. We have started the development from the original Scratch source code to create the new programming context and click capturing system. Developing on top of this environment has involved a considerable effort in time due the ActionScript programming language requires compilation and then uploading it to an online test server. These two steps are time-consuming and imply that the tests are much slower. We hope that the development of the tool and its adaptation will be faster with the appearance of the web version of Scratch. In this scenario, the language will be interpreted, and it will not require compilation or server uploading. In terms of impact on the research project, it will only involve making some minor changes to the software architecture layer in order to capture student interactions in click format.

The solution has been tested in different Scratch workshops held at La Salle Campus Barcelona during 2017/2018. Due to the limited time available from the start of the project until the first workshop, we have been able to develop a stable version, although there is still a long way to go to complete all its possibilities. However, enough interactions have been extracted to offer a first analysis and results of the tool.

We can differentiate four stages of development with their consequent technological systems and architectures. The first stage sets the software architecture necessary to capture the clicks. The second one establishes a web service to receive and store data. We have called it Datagate. The third stage proposes an initial predictive model in relation to the data captured in the previous stage. The fourth stage presents additional visualizations to integrate with visualization tools such as Students Progress Snapshot to help teachers in their task as evaluators and tutors.

Figure 2 shows the flow of technologies used for the development of the modified Scratch tool with Learning Analytics which is explained in the next sections.

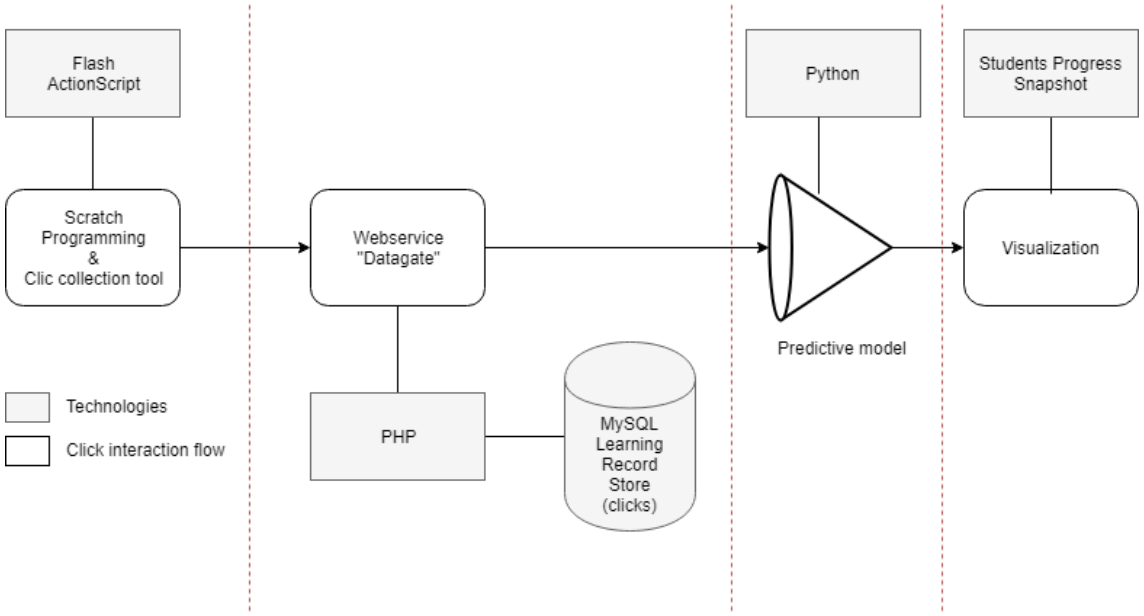

Figure 2. The flow of technologies used for the development of the modified Scratch tool.

The software architecture phase involved the analysis and understanding of the source code of Scratch. Scratch is an open source, downloadable and modifiable tool developed in Flash technology. In the project, we have used the code to add a new logical layer that allows collecting all clicks on the user interface. In this way, we have not modified the original functionalities of the code and therefore our tool maintains all initial features.

Our logical layer is based on a software architecture model developed specifically to capture the clicks on the user interface. The Flash development environment works by events so that each student click generates a series of internal messages with associated information. This information is composed, among other data, of the vertical and horizontal position of the click on the screen, which object has been clicked and its container object. The designed capturing model manages to collect each click event and extract additional information from the clicked object such as its name, stored text, object to which it belongs, and other characteristics that allow it to be identified within the Scratch development environment.

All this extracted information allows to know if the student has clicked on a specific block of Scratch, on a particular button, on the scripts development space or sprites designer or also if he has dragged a block to a particular zone of the user interface or has stacked it under another block of their programming algorithm or even if the student has rearranged the blocks within their algorithm or deleted one or more blocks. With this information, we can identify behavior patterns such “slow and analytical programmer” or “programming based on rapid cycles of trial and error”.

Scratch runs in an online environment. This fact implies that the information generated in the click-capture model must be sent and stored in a server outside the student's computer. This server acts as a learning record store. Therefore, the modeling architecture design of the first stage considers sending the data associated with the clicks outside the student's computer through a web service.

This second stage involves the creation of the web service. We've used PHP and MySQL technologies to enable communication between the click capture model and the learning record store. This provides the ability to save every data generated by a click on Scratch in a database accessible afterward for analytics purposes. We call this web service internally Datagate since it is the gateway to the collection of data and at the same time the input for the analysis stage.

As said in section 3.1, this research is a follow-up to the previous work [1]. The technology used was PHP and MySQL due to the architecture of the virtual learning environment used as the source of educational data. We considered using the same technology to enable compatibility with the previous work. The Flash technology was used due to the Scratch development environment is based on this technology. We would have preferred to use the HTML version of Scratch since it offers much more versatility in data collection through different devices, but according to official sources, this version will not be available until January 2019. A first preview will be ready for this August 2018, version that we will use for the following data collection.

The use of the Python programming language was a decision based on the availability of statistical and machine learning libraries, as well as compatibility with PHP and MySQL technologies. PHP is not a language in which its strength is to develop machine learning applications. In addition, there are limited libraries still in development that could bias or limit the development of work. For example, there is the PHP-ML library with statistical operations with possibilities to calculate the Pearson correlation coefficient but not the Spearman. For this reason, we choose to use Python to apply machine learning algorithms and for the predictive model. In addition, from PHP you can use Python results or even execute Python scripts. The modified Scratch application is loaded and displayed inside a browser opened in the student's computer. For every and each interaction based on clicks, it communicates with this web service to store data related in raw format.

The next stage will allow the analysis of the stored learning records in two senses. First, it is intended to identify in the data the different categories of students proposed later in this work. Second, it is intended to visualize this information through a quantitative approach.

3.3 Analytics and predictions

Fields et al. [12] show in their research how it is possible to use learning analytics to understand how students program in Scratch. The study is based on the analysis of the time dedicated and the developed code of the solutions to the proposed problems. Their results show different metrics in relation to application initialization and concurrent execution. We think that a new approach based on clickstream is possible to discover new behaviors.

We consider at the same time that the analysis must be accompanied by an evaluation of the teaching staff to establish a relationship between qualifications and behaviors. This relationship will allow teachers to obtain information on possible risks and act accordingly.

3.3.1 Rubrics

In the Scratch workshops for students and teachers, we used a rubric shown in Table 1 to assess specific aspects of programming activities. With the analysis of the collected data, we intend to automate this rubric in some way so that the evaluation could be in real-time and the teacher could know the status of the student to offer high-quality real-time feedback [4, 19]. We hope to generate new rubrics in a near future to evaluate other aspects of programming and find new predictive models.

Table 1. Rubric for programming assessment

Aspects to assessment	4	3	2	1
Initialization	Initializes variables when the green flag is pressed.	Initialize some of the required variables.	Initialize some of the required variables incorrectly.	It does not initialize variables when pressing the green flag.
Loops	Uses different types of loop structures when necessary.	Uses some but not all types of loop structures.	Uses a single type of loop structure to solve all casuistics.	Does not use loop structures.
Conditionals	Use conditionals with and without alternatives when necessary.	Only use conditionals with alternative.	Use only conditional without alternative.	It does not use conditionals.
Variables	Use variables correctly when necessary.	Use some variables correctly but not necessary.	It uses some or too many variables incorrectly.	It does not use variables.
Communication between objects	Use messaging to communicate between objects.	Use variables to communicate between objects and some message.	Use only variables to communicate with objects.	It does not perform communications between objects.

3.3.2 Behavior patterns

To detect and predict behaviors we incorporate machine learning and big data techniques. In a Big Data environment data grows at a very fast speed, is very varied and has a considerable volume. This defines what is known as the 3VS [42], or the essential characteristics of a Big Data environment. In our research to help teachers to take the best decisions and improve their students' learning, we had committed to visualizing data to help achieve it. This work understood as the continuation of the previous one in which the progress of a student is shown in a visual format, aims to give teachers a more detailed vision of their students. In this sense, we began to deal with a large amount of data of all the student's interactions in the virtual learning environment and now in the interactions in visual programming environments. Our work continues to deepen aspects of visualization and, therefore, continues with the collection of additional data in parallel to those that are already collected. The aim of this work is to complement, optimize and improve the accuracy of the data offered so that teachers can have quality, reliable and safe tools in terms of data collection and visualization. That is the reason why we are concurrently researching in university environments to emulate the present research in non-visual environments. In addition, steps are being taken in the application of emerging technologies such as Blockchain to ensure protection and security of data and identities of students in data collection in learning analytics processes.

We continue looking for workshops to transfer the challenges to real classes in educational stages of primary and secondary school throughout different schools. All this information will become in little time from linear to an exponential with the huge amount of data what we will get. In this sense, we focus our work on the use of big data techniques and, in particular, of the a priori knowledgeable learning machine, that we have a data environment in which we cannot draw conclusions with manual analysis, soon we are faced with an environment of considerable dimensions in which we must act with methodologies of discovery, prediction, and modeling.

At the beginning of the investigation we did not know the implications that a work of this style could have, now we see that if we have been able to extract patterns in an environment with considerable data collected, we hope to offer new improved possibilities when we have much more data. It is for this reason that we consider using supervised approaches in the application of machine learning algorithms. Using machine learning allows us to explore, discover and automate models in the available data. Within this approach,

two scenarios are defined that are differentiated by the availability of data, which are supervised learning versus unsupervised learning. The supervised algorithms are used in environments where data is already available. These are used to enter the algorithms with the aim of offering specific results as soon as new data is introduced. In our investigation, this is a step that will take place shortly since we already have data and we can train the initial models to offer better predictions and patterns of more precise behaviors. Using this type of machine learning algorithms are helpful to complement our SPS dashboard.

For a first viewing of the data, we apply clustering techniques that allow grouping students according to the zones of Scratch user interface they have clicked on. Of the two approaches, the supervised and the unsupervised, we use the second because when we started to capture data, we had an unknown environment in which we needed to uncover possible patterns of behavior. This is the reason why we use clustering algorithms, to group similar behaviors and start drawing conclusions from the data we had available. This technic gave us the first map of clustered clicks from which to begin to understand possible behaviors of the students.

To understand the concept of "map of clustered clicks" we present in the Figure 3 the interface of the Scratch program used by students.

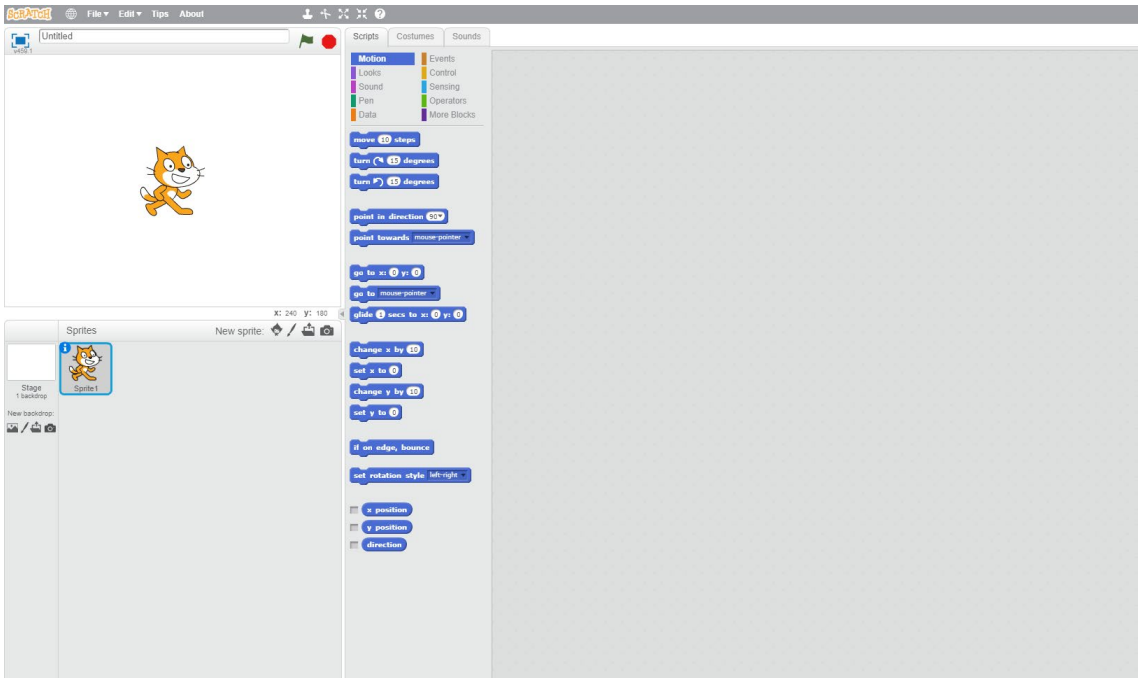

Figure 3. User interface of Scratch.

Figure 4 shows all the clicks made by all students who have been in any of the Scratch workshops. As it can be seen, the clicks are made all along the interface. The user interface of Scratch and its parts can be extrapolated from the image.

Figure 4. Map of clicks made by all students who have been in any of the Scratch workshops.

This visualization was not enough to begin to extract possible behaviors. Therefore, we conducted a clustering process based on k-means and 7 clusters. This analytical process resulted in a very interesting click map as shown in Figure 4. From this new map of clicks, we could begin to draw real conclusions about the behavior of the students and propose the first patterns of behavior in programming activities.

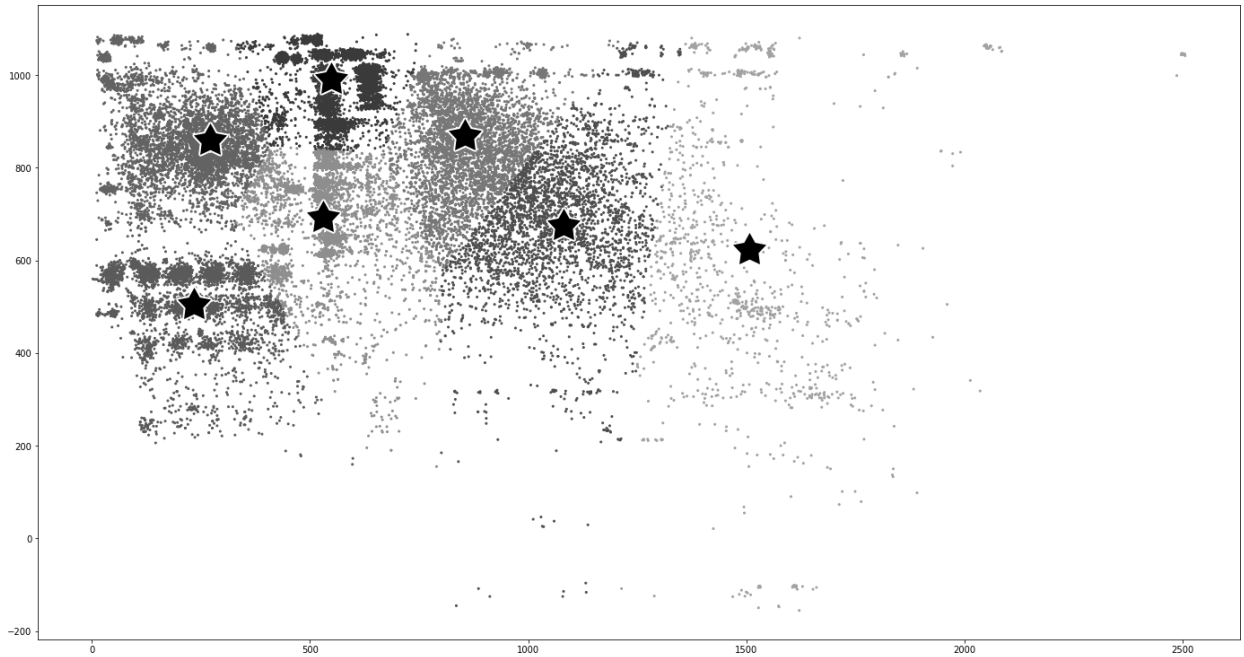

Figure 5. K-means of 7 clusters of clicks made by all students who have been in any of the Scratch workshops.

Figure 5 shows 7 clusters identified by a star. This separation by groups of clicks offers us a first understanding of the behaviors. Those students whose majority of clicks are in the three clusters on the right of the image could be called "students more prone to development" due they click on the part of Scratch where the development blocks are inserted. The students which their clicks are in the first two clusters starting from the left of the image could be considered "students more prone to design" due it is the Scratch area in which the different sprites are created and positioned. Hence, we can create a first taxonomy if students' behavior related to the concentration of clicks:

- **Students more prone to development**: Students focus the clicks on the scripts space where the blocks are released and in the part of blocks available to be dragged to the scripts space.
- **Students more prone to organize**: Students focus to move rather than add, modify or delete objects, blocks or sprites.
- **Students more prone to design**: Students focuses the clicks in the area of sprites and scenario.
- **Students more prone to create multimedia**: Students focus the clicks in the area of sounds and drawing tools.

This first classification is very useful for teachers so in a first place they can detect which students are spending more time in some areas not really needed in the assignments. In the results section, we show the different visualizations created for the SPS dashboard.

Afterward, we created a sole image result of the fusion of Scratch interface and the first map of clicks. We were able to discover then new patterns of behavior with which to relate possible risks in the delivery of activities.

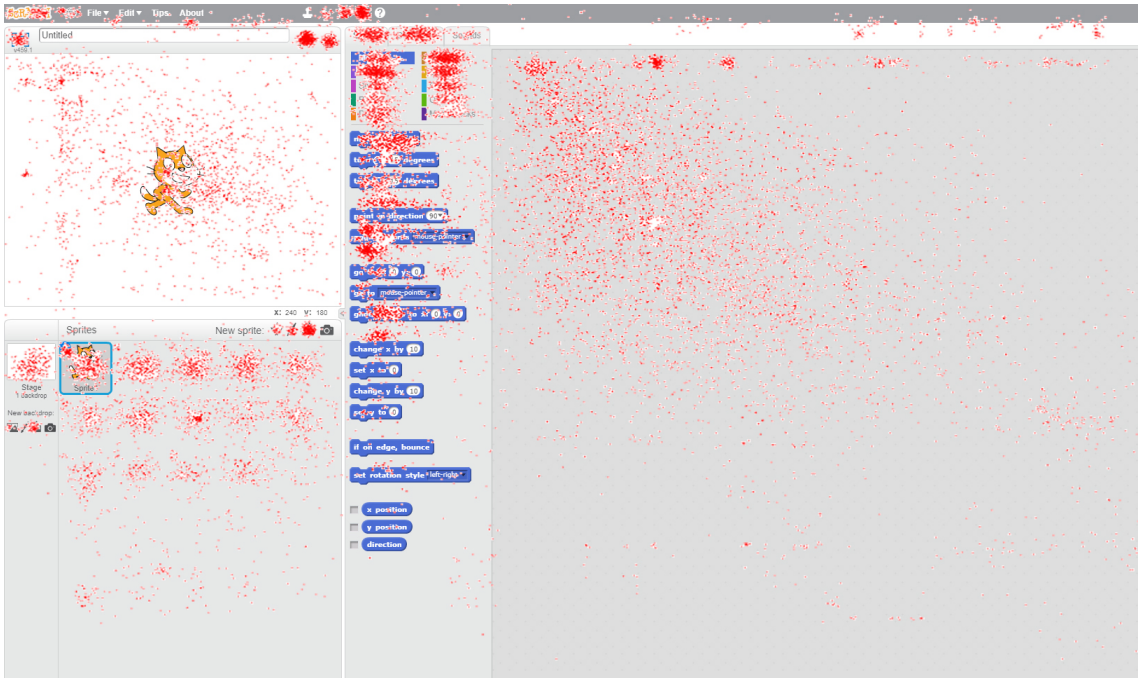

Figure 6. Fusion of Figure 3 and Figure 4.

In the image above (Figure 6) is shown exactly where students click on Scratch. It can be seen that students interact a lot with the size of the sprites and how students interact with the "green flag" buttons and the "stop" button. These two buttons are used in Scratch to indicate where the code begins execution or to initialize the status of the program.

The analysis of the use of these buttons can be associated with rubrics evaluation and offer useful information to teachers. We wanted to detect the following patterns analyzing the clickstream of these two buttons:

- **Blocked development (blocked)**. This type of behavior defines those students who are behind in the coding of the program. The motives can be different, for example by distractions, by attention to non-coding aspects such as design and graphics or by a true ignorance of how to code the solution.
- **Development at a normal (normal) pace.** This type of behavior defines those students who have a balance of development between the graphics interface of the program and the coding.
- **Rapid development based on trial-error (rapid)**. This type of behavior defines those students who make rapid changes in the program and constantly check the results.

We could have proposed many more patterns for automatic detection. However, we want to limit the research to reaffirm the possibilities of clickstream in the detection of types of behavior, validate the predictive model and incorporate more metrics to improve its scope and accuracy.

The discovery of these patterns is based on the number of clicks made on the "green flag" button. Scratch allows you to execute the developed code by pressing a button identified with a green flag. This action allows you to indicate in which part or parts of the code it must start the execution, the initialization of the variables or the initialization of the program status and to provide a structured execution flow. These aspects are those that are evaluated in the rubric and those that are intended to associate to the behavior patterns during development of programming activities.

For the analytical detection of the three types of behavior presented -blocking, normal, fast- we have made a statistical approximation. This allows us to measure the pace of development of the projects in relation to the class group.

The interactions between the different groups flow spontaneously, which changes their way of developing and results since they exchange knowledge throughout the development. Therefore, in the statistical analysis, all the clicks of all the groups of each workshop are considered. Consequently, to extract the types of behavior we elaborate a statistical calculation in relation to the median of the clicks made on the green flag button.

The predictive model of behaviors is summarized in the following three points:

- Groups that click on the green flag below the 10% of the average are considered **blocked developments**.
- Groups that click on the green flag above the 90% of the average are considered **rapid developments based on trial and error**.
- The other groups are considered as **development at a normal pace**.

In the analysis of the collected data, we found a relationship between the results of the rubrics and the patterns detected. A close relationship between results and patterns will help teachers personalize student learning. For example, a low qualification in the rubric should be reflected in a blocking behavior. In some way, this approach serves as an automation of the rubric through student behavior. This information can be obtained in real time. Consequently, the teacher may act before the student turns in the activity.

3.4 The sample

The sample comes from a series of workshops held at La Salle Campus Barcelona. La Salle offers a series of workshops related to robotics, programming or engineering thinking to all its schools in Catalonia and other non-profit organizations such as foundations that offer scholarships to high school students with high capabilities. The workshops we offer have a format of 2 to 4 hours in which we introduce theoretical concepts and place collaborative challenges for the participants. The activities are collaborative and students make groups of 2 or 3 people.

We needed a sample where students did not have computer programming knowledge or an elemental practice, and a sandbox-programming context where they could feel free and fearless to develop their own solutions in a try-error approximation. These workshops were suitable for the research due students were not experienced on programming and in conjunction generated raw data about their interaction with Scratch through clickstream.

This research analyzes the data obtained from all the Scratch workshops offered to students of different ages between 15 and 17 years old (58 students, age average: 15.96, standard deviation: 0.35). The Scratch workshop format consists of a theoretical introduction of how to approach projects in this visual programming environment. Next, the challenge of creating a Scratchroom is exposed. A Scratchroom aims to simulate a real escape room, or what is known as an "escape game". An escape room is a physical adventure game in which players solve a series of puzzles and riddles using clues, hints, and strategy to complete objectives at hand. The main objective is to escape from the place where they are locked. The participants of the workshops must create the game scenarios, the players and objects in sprites format, the sounds and develop the computational logic behind each action to offer a playable and entertaining environment. The challenge is understood as an activity of open programming, where the result depends on the ingenuity and programming abilities of the students. We can perfectly evaluate different aspects such as initialization of the environment and variables, use of repetitions of programming structures, use of computational logic by arithmetic, comparator and boolean operators. As well as the concept of concurrency or parallel execution of instructions by the control of variables or by Scratch's own methods such as messaging between objects. In short, the data sample is extracted from all the clicks made by groups of 2 to 3 students in specific workshops of open resolution programming challenges with Scratch.

Modifications in the Scratch source code have allowed capturing more than 38,000 clicks made by students among more than 20 different programming projects. The data collected includes the position of the click on the screen, the date and time of the moment of interaction, the object in which the click was made, the container of the object and the action taken, such as moving, adding or deleting blocks.

3.5 Analysis

This dataset has made it possible to identify quickly and without too much in-depth analysis some of the expected behaviors in the development environment. First we conducted an exploratory analysis through histograms, basic scatter plots and advanced analytics to understand the sample and extract firsts results. Afterwards, we conducted a bivariate correlation to measure how the behaviors applied to the rubrics grade. A first analysis based on the clustering technique based on k-means has been able to produce a graph in which the clicks are identified, and different possible types of students are visualized. Each red star identifies a cluster of clicks. The students who concentrate the clicks in the lower right, identified by the yellow or lilac color, are those that fall into the Developer category and therefore pay much more attention to coding than to other more artistic aspects.

In regard to the bivariate correlation, and in a very first step, we visualized the relationship between the variables. The result was a scatter plot showing a positive relationship in terms of the variables "clicks on the green button" and "results of the rubric". Therefore, the linear relationship showed two trends: the more clicks, the better the results in the rubric, and the fewer the worse clicks re-cluttered. This is the reason why we performed a Pearson correlation, instead a Spearman one, since the variables already had a linear rather than a monotonic relationship in the first visualizations.

4 Results and discussion

The projects developed in the workshops have been evaluated by the two instruments proposed in this work. On the one hand, the assignments' solutions have been evaluated with the rubric elaborated in previous sections. On the other hand, clicks have been introduced into the predictive model to extract behaviors. This has allowed to obtain a score and a taxonomy for each of the projects and find the first correlation between them. Table 2 shows the results of some of the more than 20 projects assessed in terms of the rubric and behavior. The data exposed is a subset of the total of clicks. The results are based on all the projects assessed by teachers, more than 20, not only considering the sample shown. The sample considers

8 groups of 2 or 3 students that suppose at least 16 students who generated those 292 clicks. There are more than 765 clicks in the learning record store made by more than 70 students, so the results are based in those 765 clicks done on the green flag button.

Moreover, the total number of students is equal or greater than the analyzed by Pathan et al. [26] and Fields et al. [12] with 70 and 64 students each one in their studies.

The total of clicks to the green flag button represents 2% of the total of collected clicks. In comparison, it may seem like a small sample but is enough to say that is useful and valid for our initial purpose. Moreover, this 2% data scenario increases our expectative to extract more useful patterns of behavior from the 98% remaining.

Table 2. Some of the projects assessed by rubrics and behavior

Project	Rubric grade	Clicks on green flag button	Behavior
Group #1	13	83	Rapid
Group #2	10	83	Rapid
Group #3	12	66	Rapid
Group #4	5	1	Blocked
Group #5	13	52	Rapid
Group #6	5	1	Blocked
Group #7	6	2	Blocked
Group #8	9	4	Blocked

Results of the analysis of the bivariate correlation of the rubric grades and the clicks on the green flag button are shown in Table 3.

Table 3. Results of the analysis of the bivariate correlation considering rubric grades and clicks on the green flag

p-value	Correlation coefficient
0.02380621256650697	0.926184147506773

As observed in Table 3, the correlation coefficient is close to 1. This indicates a strong correlation between the evaluation by rubric and the clicks to the green flag button. The p-value of the statistical test is less than 0.05. This indicates that there is a statistical significance. Consequently, it can be affirmed that there is a correlation between those projects in which students make rapid iterations of trial and error and good results. On the other hand, the students who make fewer executions of the program are those who have the simplest or unfinished programs.

These results provide a scenario that is very useful for teachers. We can create visualizations of the behavioral trends before students' deliveries, so teachers can detect risks and act accordingly to avoid

failures in the activities. Figure 7 shows the frequency of clicks on the green flag button. This exposes the behavior of the students in relation to the execution of the code.

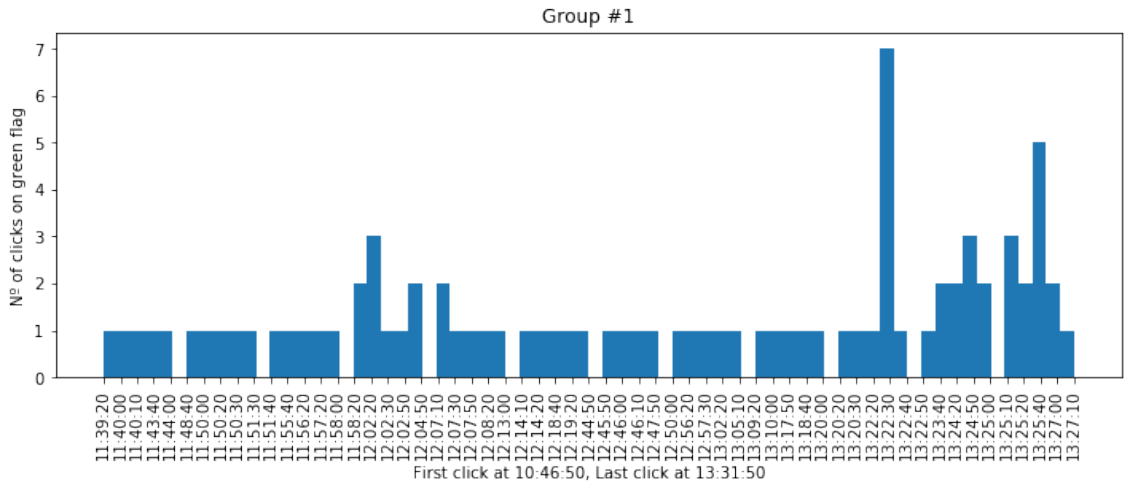

Figure 7. The frequency of clicks on the green flag button by group 1.

Figure 7 histogram shows how a group of students categorized as rapid development is continuously running the code with some breaks of 3 or 5 minutes. On the other hand, Figure 8 represents another histogram that shows how group number 8 performs very few clicks. It can be seen how they have occupied their most time in non-development tasks and tried to perform the coding in the last 5 minutes of class. This behavior is classified as blocked development and presents a high risk of failure. In this type of behavior, the faculty will have the opportunity to urge them to put on the development or to apply the solution that they think is most convenient.

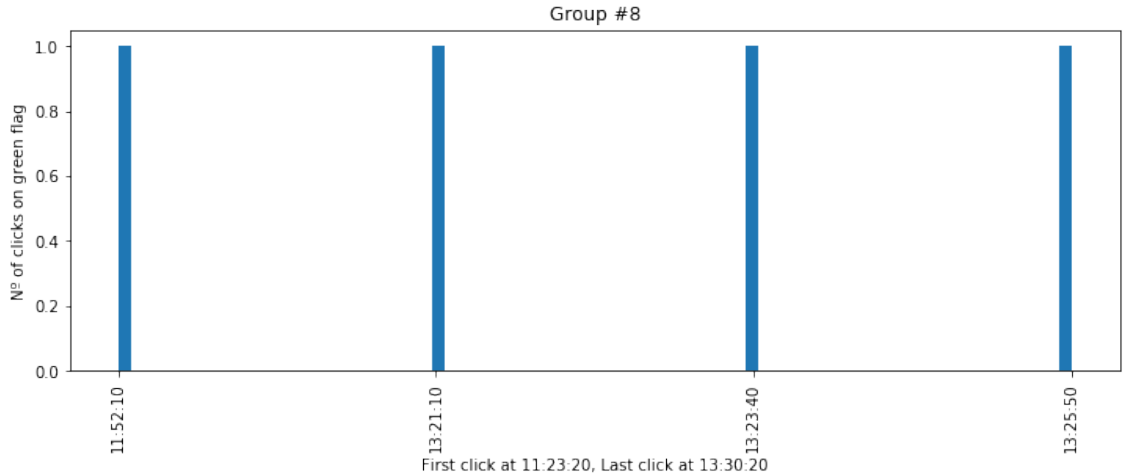

Figure 8. The frequency of clicks on the green flag button by group 8.

Both these histograms and the results of the predictive model can be easily incorporated into what we call Students Progress Snapshot. In this visual space, the teacher integrates all the possible visualizations of the

student's learning state. Therefore, from one place you can show each professor the histograms of each student and the predictors according to their risk status.

We also added new visualizations into the SPS to deliver better insights to teachers. In Figures 9, 10 and 11 teachers can see and easily understand the detected behaviors in the predictive analysis. With colors, basic shapes and different positions a teacher can detect students with blocked development behavior if the circle is orange and left position, with development at normal pace behavior if the circle is black and centered or with rapid development based on trial-error behavior if the circle is green and right position.

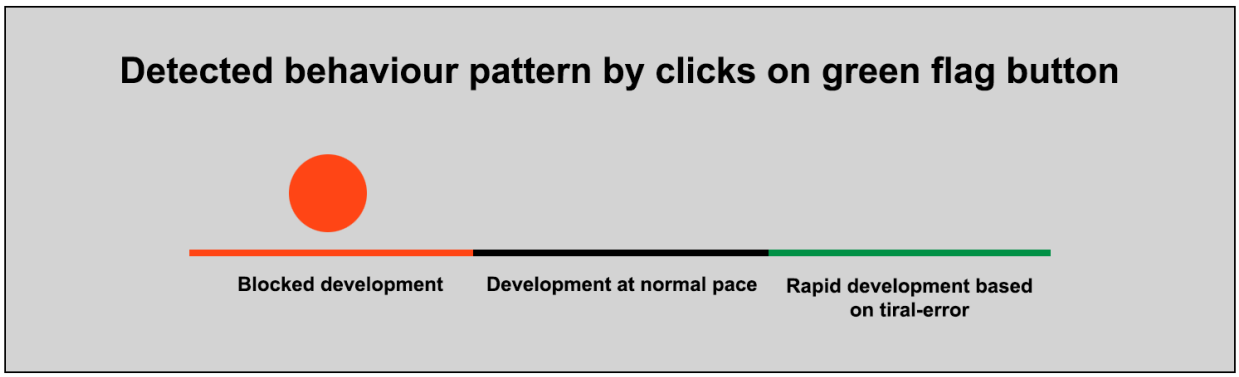

Figure 9. Detected "Blocked development" behavior pattern by clicks on green flag button.

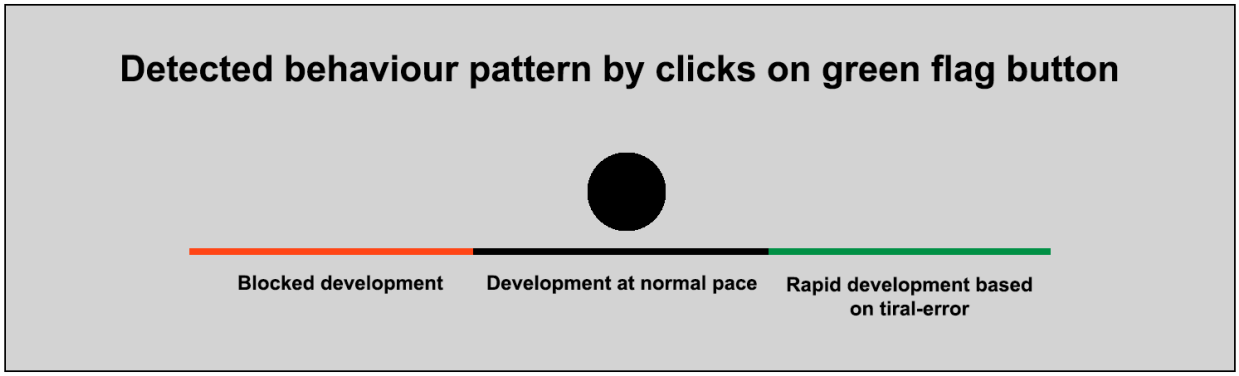

Figure 10. Detected "Development at normal pace" behavior pattern by clicks on green flag button.

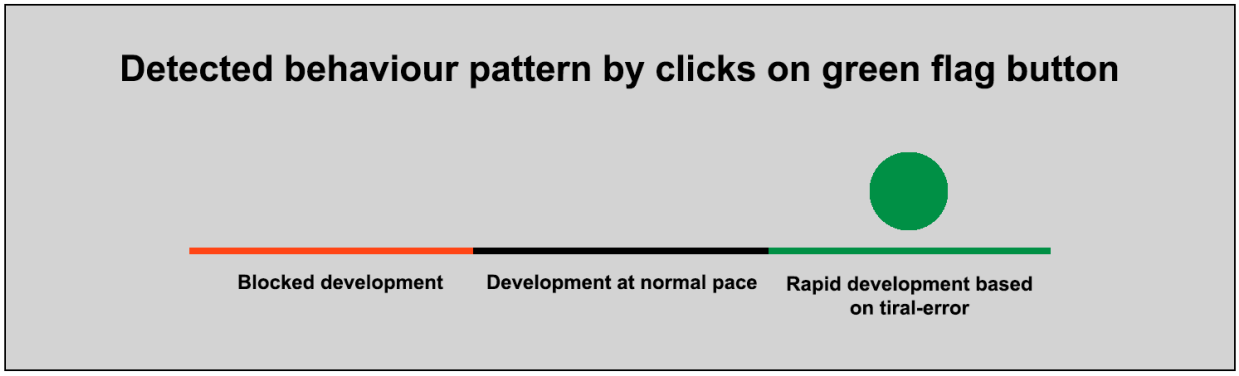

Figure 11. Detected "Rapid development based on trial-error" behavior pattern by clicks on green flag button.

In Figures 12 and 13 teachers can discover at a glance the type of student in relation to the concentration of clicks in different parts of the Scratch environment. Through different colors, teachers can see if a student is more coder, designer, multimedia or organizer oriented or a mix of those four behaviors. The colors are sorted by relevance, for example as we can see at Figure 12, starting with dark green (Coder), light green (Organizer), light orange (Designer), and dark orange (Multimedia), or at Figure 13, starting with dark green (Designer and Multimedia), light green (Organizer), and dark orange (Coder).

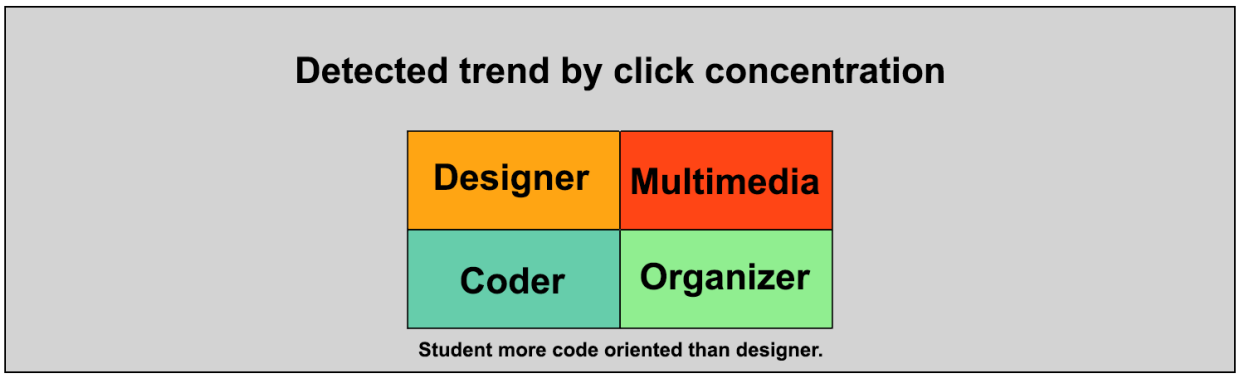

Figure 12. The detected trend by click concentration where the student is more code-oriented than a designer.

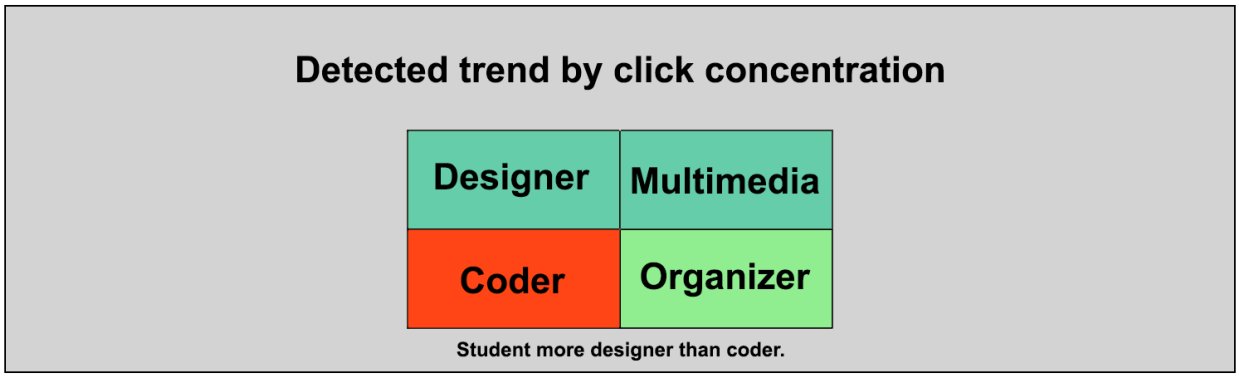

Figure 13. The detected trend by click concentration where the student is more designer than a coder.

5 Conclusions

5.1 Theoretical and Practical Implications

Until now, no research has used a full click-based technique to identify patterns of behavior within the Scratch programming environment. The different publications found address the problem by analyzing the

source code developed by the students. This technique opens new research approaches to understand how students learn and interact with Scratch in programming activities.

The results of the work mean a great advance in the field of evaluation of behavior in the programming practice. Addressing the analysis with clicks allows the recognition of students at risk and act accordingly. The use of clicks complements the evaluation of the activities delivered. Instead of analyzing the final result it is possible to check how students interact with the different elements of the programming interface. Therefore, it is possible to check positive behaviors or possible blockages.

The clicks provide data on the actions of the students, the places where they interact most and their development methodology. We hope to provide much more data as soon as the investigation has more progress.

As stated in the introduction, there are some studies similar to our research in regard to the extraction of patterns of behavior of students in programming environments. Sherman and Martin [24] propose an analytical approach to extract patterns of behavior in student developers of App Inventor projects. Other authors [19] have proposed to understand how novice programmers approach their first lines of code in conventional programming environments, through keystrokes and code analysis. Blikstein et al. [28] used EDM techniques and learning analytics in student code snapshots. All authors used code analysis to extract results and consequent conclusions.

Our study differs from the above at a methodology level, where data collection from clickstream reveals new paths of research. Our proposed approximation based in a clickstream technique only appears to be equal to studies conducted by Eguiluz et al. [11]. Nonetheless, Eguiluz et al. do not deep in the detail of clicks. In this sense, they use the clickstream to analyze the blocks that students have used, which means that they analyze the code resulting from clicking on several blocks of programming as the others studies.

Our remarkable approximation facilitates the teacher's understanding of student’s behavior in their assignments. This knowledge can impact positively in students due teachers can use it to bring quality real time in class support and feedback. We are aware that our solution is teacher focused. This draw a future line of improvement to transform the results of our research into solutions for students to help them self-improve their learning through the discovery of new behavior patterns.

We also hope to expand the behavior patterns with a deeper analysis of the clicks beyond the green flag. Other behaviors can be differentiated through the analysis of adding, modifying or eliminating blocks in the sprites coding space. This will be possible as soon as we have more data.

5.2 Limitations and Future Work

There are limitations linked to the original Flash technology that does not allow to advance at the desired pace, due to high development costs, times and other involved technologies. Flash is not available on mobile devices, which discards a large percentage of participating schools in the long-term testing phase. We hope to be able to make an exhaustive follow-up course with the same students to validate and improve the accuracy of the prediction model. This will be possible when the new version of Scratch based on HTML5 is published due will be compatible with most devices used in education. Hence, we can check how students develop in different devices and educational situations as well.

Another limitation is the difficulty of storing the multimedia information of Scratch. As a result, no audio, video or image of student developments is currently stored. This implies that the state of the program can't be visually reconstructed at any given time. We believe that this visualization would provide more data to teachers and we hope to overcome these limitations in the third version of Scratch, at which time HTML5 will be used as a development language.

5.3 Final Considerations

Scratch is an environment that allows students to develop their programming skills with an agile language, based on blocks, interactive and safe. It is a visual programming environment understood as a sandbox where students can do and undo as their will while they test what have learned in class and evolve their algorithms and programming techniques. In these environments students require the teacher's help to advance safely, to develop their programming skills and not to lock themselves in the development of projects. In addition, a teacher cannot scale itself to review in real time all the work of students and thus give a quality feedback. The reason is that analyzing all the code generated by all the students requires a great effort and time. Our work gives all teachers who use Scratch in their classroom the ability to scale, review and give quality feedback in real time.

This work has been possible thanks to the release of the Scratch source code. It has been possible to create a fork and add new software architectures that allow capturing every click of student interaction. The

analysis of the collected clicks allowed to extract different students' behaviors that helps the teacher to understand what is going on during the development of assignments, activities or projects. A first exploratory analysis revealed behaviors related to the tendency of the student in the use of some blocks or parts of Scratch instead of another one. Such analysis reveals if a student is more development-focused than design. A predictive analysis correlated with real evaluations of teachers shows how it is possible to analyze students' behaviors in relation to the clicks made in the "run" button of Scratch identified by a green flag icon. This analysis is very helpful to detect low and high-performance students during the development of assignments, so teachers can support them in real time.

The taxonomy of resulting behaviors and visualizations is very valuable for teachers, who will be able to personalize learning and improve the teaching environment. Now teachers can act before students deliver incorrect or at least incomplete tasks.

The investigation continues its course in two ways. We continue tracking students in visual programming environments but now we are doing a follow-up of the students a longer time, so we can track their evolution. This will allow us to identify new behaviors and provide new findings to enhance tutoring, following-up, and evaluation of students.

At the same time, we are transferring the results and techniques used in this work to university programming subjects to track students in non-visual programming environments to compare results. We are translating this experience into non-visual programming environments such as command line in C language projects. The procedure will be applied to the subject "Programming methodology and technology" at La Salle Campus Barcelona where students compile code in C language. Each compilation is understood as a click on the green flag button. Such association is key to evaluate the potential of the proposed technique. The system logs automatically the total number of compilations for each student and additional information such as time for each compilation and idle time in between. We hope to get relevant results during the next course.

6 Ethics

The data captured in no case store sensitive or personal information of students. Any possible sensitive data has been depersonalized since the click capture. In this way, any student interaction reaches the analysis section absolutely anonymized.

7 Agreements

This research work has been carried out within Education in Knowledge Society PhD Programme of the University of Salamanca [15]. This work is also supported by the Spanish Ministry of Economy and Competitiveness throughout the DEFINES project (Ref. TIN2016-80172-R) and the support of the Secretary of Universities and Research of the Department of Enterprise and Knowledge of the Generalitat de Catalunya with help 2017 SGR 934.

8 References

[1] Amo, D. et al. 2018. The Student's Progress Snapshot a Hybrid Text and Visual Learning Analytics Dashboard. *The International Journal of Engineering Education*. 34–3, Decision Making in Engineering Education using Learning Analytics (2018), 990–1000.

[2] Berland, M. et al. 2014. Educational Data Mining and Learning Analytics: Applications to Constructionist Research. *Technology, Knowledge and Learning*. 19, 1–2 (Jul. 2014), 205–220. DOI:https://doi.org/10.1007/s10758-014-9223-7.

[3] Berland, M. et al. 2013. Using Learning Analytics to Understand the Learning Pathways of Novice Programmers. *Journal of the Learning Sciences*. 22, 4 (Oct. 2013), 564–599. DOI:https://doi.org/10.1080/10508406.2013.836655.

[4] Blikstein, P. et al. 2014. Programming Pluralism: Using Learning Analytics to Detect Patterns in the Learning of Computer Programming. *Journal of the Learning Sciences*. 23, 4 (Oct. 2014), 561–599. DOI:https://doi.org/10.1080/10508406.2014.954750.

[5] Blikstein, P. 2011. Using learning analytics to assess students' behavior in open-ended programming tasks. *Proceedings of the 1st International Conference on Learning Analytics and Knowledge - LAK '11* (New York, New York, USA, 2011), 110.

[6] Brainerd, J. and Barry, B. 2001. *Case Study: E-Commerce Clickstream Visualization*.

[7] Chatterjee, P. et al. 2003. Modeling the clickstream: Implications for web-based advertising efforts. *Marketing Science*. 22, 4 (2003), 520–541.

[8] Costa, E.B. et al. 2017. Evaluating the effectiveness of educational data mining techniques for early prediction of students' academic failure in introductory programming courses. *Computers in Human Behavior*. 73, (Aug. 2017), 247–256. DOI:https://doi.org/10.1016/J.CHB.2017.01.047.

[9] Dale Wilson, R. 2010. Using clickstream data to enhance business-to-business web site performance. *Journal of Business & Industrial Marketing*. 25, 3 (Feb. 2010), 177–187. DOI:https://doi.org/10.1108/08858621011027768.

[10] Dyke, G. 2011. Which aspects of novice programmers' usage of an IDE predict learning outcomes. *Proceedings of the 42nd ACM technical symposium on Computer science education - SIGCSE '11* (New York, New York, USA, 2011), 505.

[11] Eguiluz, A. et al. 2017. Exploring the progression of early programmers in a set of computational thinking challenges via clickstream analysis. *IEEE Transactions on Emerging Topics in Computing*. (2017).

[12] Fields, D.A. et al. 2016. Combining Big Data and Thick Data Analyses for Understanding Youth Learning Trajectories in a Summer Coding Camp. (2016). DOI:https://doi.org/10.1145/2839509.2844631.

[13] Filva, D.A. et al. 2014. Google analytics for time behavior measurement in Moodle. *2014 9th Iberian Conference on Information Systems and Technologies (CISTI)* (Jun. 2014), 1–6.

[14] Frhan, A.J. 2017. Website Clickstream Data Visualization Using Improved Markov Chain

Modelling In Apache Flume. *MATEC Web of Conferences* (2017), 4025.

[15] García-Peñalvo, F.J. 2014. Formación en la sociedad del conocimiento, un programa de doctorado con una perspectiva interdisciplinar. *Teoría de la Educación. Educación y Cultura en la Sociedad de la Información*. 15, 1 (2014), 4–9.

[16] Grover, S. 2017. 11e Unlocking the Potential of Learning Analytics in Computing Education. (2017). DOI:https://doi.org/10.1145/3122773.

[17] Grover, S. et al. 2017. A Framework for Using Hypothesis-Driven Approaches to Support Data-Driven Learning Analytics in Measuring Computational Thinking in Block-Based Programming Environments. *ACM Transactions on Computing Education*. 17, 3 (Aug. 2017), 1–25. DOI:https://doi.org/10.1145/3105910.

[18] Ihantola, P. et al. 2015. Educational Data Mining and Learning Analytics in Programming. *Proceedings of the 2015 ITiCSE on Working Group Reports - ITICSE-WGR 15* (New York, New York, USA, 2015), 41–63.

[19] Kaur, P. et al. 2015. Classification and prediction based data mining algorithms to predict slow learners in education sector. *Procedia Computer Science*. 57, (2015), 500–508.

[20] Koh, K.H. et al. 2014. Real time assessment of computational thinking. *2014 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)* (2014), 49–52.

[21] Learning Analytics and Educational Data Mining | Erik Duval's Weblog: *https://erikduval.wordpress.com/2012/01/30/learning-analytics-and-educational-data-mining/*. Accessed: 2018-04-24.

[22] Lee, J. et al. 2001. Visualization and Analysis of Clickstream Data of Online Stores for Understanding Web Merchandising. *Data Mining and Knowledge Discovery*. 5, 1/2 (2001), 59–84. DOI:https://doi.org/10.1023/A:1009843912662.

[23] Lye, S.Y. and Koh, J.H.L. 2014. Review on teaching and learning of computational thinking through programming: What is next for K-12? *Computers in Human Behavior*.

[24] Martin, T. and Sherin, B. 2013. Learning Analytics and Computational Techniques for Detecting and Evaluating Patterns in Learning: An Introduction to the Special Issue. *Journal of the Learning Sciences*. 22, 4 (Oct. 2013), 511–520. DOI:https://doi.org/10.1080/10508406.2013.840466.

[25] Montgomery, A.L. et al. 2004. Modeling online browsing and path analysis using clickstream data. *Marketing science*. 23, 4 (2004), 579–595.

[26] Pathan, A.A. et al. 2014. Educational data mining: A mining model for developing students' programming skills. *The 8th International Conference on Software, Knowledge, Information Management and Applications (SKIMA 2014)* (Dec. 2014), 1–5.

[27] Pero, Š. 2013. Modeling Programming Skills of Students in an Educational Recommender System. Springer, Berlin, Heidelberg. 401–404.

[28] Phippen, A. et al. 2004. A practical evaluation of Web analytics. *Internet Research*. 14, 4 (Sep. 2004), 284–293. DOI:https://doi.org/10.1108/10662240410555306.

[29] Piech, C. et al. 2012. Modeling how students learn to program. *Proceedings of the 43rd ACM technical symposium on Computer Science Education - SIGCSE '12* (New York, New York, USA, 2012), 153.

[30] Rivers, K. et al. 2016. Learning Curve Analysis for Programming: Which Concepts do Students Struggle With? *Proceedings of the 2016 ACM Conference on International Computing Education Research - ICER '16* (New York, New York, USA, 2016), 143–151.

[31] Scratch Statistics - Imagine, Program, Share: 2007. *https://scratch.mit.edu/statistics/*. Accessed: 2018-08-28.

[32] Seiter, L. and Foreman, B. 2013. Modeling the learning progressions of computational thinking of primary grade students. *Proceedings of the ninth annual international ACM conference on International computing education research - ICER '13* (New York, New York, USA, 2013), 59.

[33] Senecal, S. et al. 2005. Consumers' decision-making process and their online shopping behavior: a clickstream analysis. *Journal of Business Research*. 58, 11 (Nov. 2005), 1599–1608. DOI:https://doi.org/10.1016/J.JBUSRES.2004.06.003.

[34] Shaun Joazeiro de Baker, R. and Salvador Inventado, P. 2014. Chapter X: Educational Data Mining and Learning Analytics. (2014).

[35] Sherman, M. and Martin, F. 2015. Learning analytics for the assessment of interaction with App Inventor. *2015 IEEE Blocks and Beyond Workshop (Blocks and Beyond)* (Oct. 2015), 13–14.

[36] Siemens, G. and d Baker, R.S.J. 2012. Learning analytics and educational data mining: towards communication and collaboration. *Proceedings of the 2nd international conference on learning analytics and knowledge* (2012), 252–254.

[37] Tabanao, E.S. et al. 2011. Predicting at-risk novice Java programmers through the analysis of online protocols. *Proceedings of the seventh international workshop on Computing education research - ICER '11* (New York, New York, USA, 2011), 85.

[38] Vihavainen, A. et al. 2014. How novices tackle their first lines of code in an IDE. *Proceedings of the 14th Koli Calling International Conference on Computing Education Research - Koli Calling '14* (New York, New York, USA, 2014), 109–116.

[39] Werner, L. et al. 2013. A First Step in Learning Analytics: Pre-Processing Low-Level Alice Logging Data of Middle School Students. *Journal of Educational Data Mining*. 5, 2 (2013), 11–37.

[40] Wilson, R.D. 2005. Using web traffic analysis for customer acquisition and retention programs in marketing. *Services Marketing Quarterly*. 26, 2 (2005), 1–22.

[41] Xie, B.X.-Y. 2016. Progression of computational thinking skills demonstrated by App Inventor users. (2016).

[42] 2001. *Application Delivery Strategies*.