INTERNATIONAL JOURNAL ON INFORMATICS VISUALIZATION

journal homepage : www.joiv.org/index.php/joiv

Development of Programming Log Collection System Requirements Using Interface Requirement Analysis Techniques

Huijae Park [a], Wongyu Lee [b], Jamee Kim [c,*]

[a] Department of Computer Science and Engineering, Graduate School, Korea University, Seoul, Republic of Korea
[b] Department of Computer Science and Engineering, College of Informatics, Korea University, Seoul, Republic of Korea
[c] Major of Computer Science Education, Graduate School of Education, Korea University, Seoul, Republic of Korea

Corresponding author: *celine@korea.ac.kr

***Abstract*—As software affects each industry, companies are increasingly trying to recruit talent. Despite the interest and investment of companies, it is difficult to find talents with technical expertise and specialization in a specific field at the same time. As a result, companies have begun to discover talents who can overcome their lack of expertise through talents with high problem-solving skills. Countries worldwide that felt the need to discover competitive talents began to show changes in education for nurturing talents. In particular, the results of the expansion and increase of programming education that cultivate problem-solving ability have begun to be seen. However, programming education is different from existing education, and many learners have difficulties with the introductory process due to the difficult debugging process. In order to analyze the difficulties of introductory learners and support their learning, a system that can collect data from the programming process and analyze behavior types is required. There are several methods for deriving the system requirements, but the interface requirements analysis method was selected in this study. We approached how to process data in the system by deriving the type of data that the system administrator wants to collect. This study laid the foundation for a system that can analyze the programming process of introductory learners by deriving the functional and non-functional requirements required by the data collection system through interface requirements analysis.**

***Keywords*— Computer education; log collection; requirements; scratch.**

Manuscript received 25 Oct. 2021; revised 29 Dec. 2021; accepted 11 Jan. 2022. Date of publication 31 May 2022.

I. INTRODUCTION

After the 4th Industrial Revolution, as the importance of artificial intelligence and software emerged and the impact of technology on society became stronger, various fields of the modern industry began to enter the direct and indirect influence of software [1]–[3]. Nevertheless, the complexity or importance of the software in areas where software was developed and used, the application software began to be produced, reflecting the knowledge and programming skills of the field [2]. The recent workforce market has led more companies to invest more in the workforce market than in the past, with more companies demanding professional skills in the industry that require software and those who have the skills to develop software simultaneously [1], [4]. In this trend in society, it became difficult to find talented people who knew various fields of knowledge, and companies began to demand talented people with high problem-solving skills [1], [5].

The phenomenon of a society that seeks talent with problem-solving skills can be understood as a phenomenon of a generation that needs people skilled in algorithms and computer programming [1]. The subject to develop problem-solving skills is the information computing subject, which is consistent with the core competencies of the information computing subject [6]. In order to get high scores in information computing subjects, students sometimes go through the process of developing better algorithms and programming them at high speed. As a result, it can be a good achievement for students, but it does not correspond to the competence of information computing subjects. The information computing subject aims at fostering computing thinking and problem-solving skills through programming education and cultivating a cooperative attitude [7], [8]

The evaluation method of education is shifting from outcome evaluation to process-oriented evaluation [9]. However, current programming evaluation tools only evaluate the level of requesting students' programming results and compare learning results to correct answer results. Due to

changes in society's perception, the target age group of programming education is getting lower, and education that makes programming results the same as correct answer results causes rejection of programming education.[10], [11] This educational direction means a different trend from the trend of developing problem-solving skills through programming education, and rather, learners feel a great burden in the process of learning programming. This is because the environment for programming is different from the existing method of studying literature or mathematics. Programming education in terms of opening a book, reading a book, holding a pencil, and debugging while holding a keyboard and looking at a monitor feels unfamiliar. Introductory programming learners who have encountered learning in an unfamiliar environment feel various difficulties in the trial-and-error process of solving problems [12].

Various tools to solve the visual monotony and difficult debugging process, which are one of the difficulties of introductory programming learners, began to appear. A block programming language called Scratch was created by MIT based on Google's Blockly library [13], [14], or a programming web editor that micro: bit puts on its website [16]. These programming tools shorten the debugging process by quickly showing the results of the programming process. The reduction in debugging time offsets the boring or complicated process caused by trial and error in programming [12], [15], [16].

This paper conducted a study to derive the requirements of the programming log collection system. Based on the block programming environment, lowering the entry barrier to programming and lowering the rejection of learning itself will enable collecting data from learners' pure programming processes without rejection. In order to proceed with this study, it is necessary to examine the existing research on how the existing log collection systems constructed the programming system. Programming log collection means the collection of learning processes in a broad sense. Therefore, we comprehensively investigated the development of tools for collecting logs and the research for collecting programming learning processes.

Bennedsen and Caspersen [17] videotaped the programming process to solve the difficulties learners face. Although the method conducted by this study is fundamental, it has been shown that the programming process enables learners to support learning difficulties. This study demonstrated the need to develop a flexible system that can solve this problem rather than recording and analyzing daily learners. This method may seem inefficient, but it has been maintained for a relatively long time because it is easy to access. This phenomenon is also a counterexample of the difficulty of developing a programming log collection tool [18], [19].

Boisvert [20] introduced an interface in the programming process, such as the existing screen recording tool Viewlets, to solve the analysis of the programming process through video analysis methods or learners' interviews. Boisvert performed process visualization by enhancing the ability to annotate during the screen recording of Viewlets as an interactive function. This study set the stage for establishing an interface for programming visualization, but there was a limitation in that it was difficult to use it as a tool for programming beginners due to the restriction of a text-based programming environment [21].

D. Filva and four others tracked mouse trajectories and events that occurred while learners programmed in the scratch editor. The learner's programming process was tracked by applying the motion algorithm of the UX tool, which tracks the user's mouse trajectory and represents it as a heatmap. This resulted in visualization of the programming process and representation but failed to track results such as what the learner did [22].

This study derives requirements for introductory programming learners to collect the process of programming. It is necessary to define the learner's behavior type to visualize the programming process. We also derive the functional requirements of the interface for collecting and storing programming log data.

II. Materials And Method

A. Raw Data Analysis

The system that this study ultimately aims to develop is to collect logs generated in the programming learning process to understand what learners do not understand properly and to feedback on this process to teachers. In the case of introductory programming learners in this process, most of them encounter the programming environment with scratches, so it is necessary to check the data generated during the programming process of scratches.

Scratch's programming data can be checked through a simple process [23]. Download the scratch project file as shown in the leftmost file in Fig. 1.

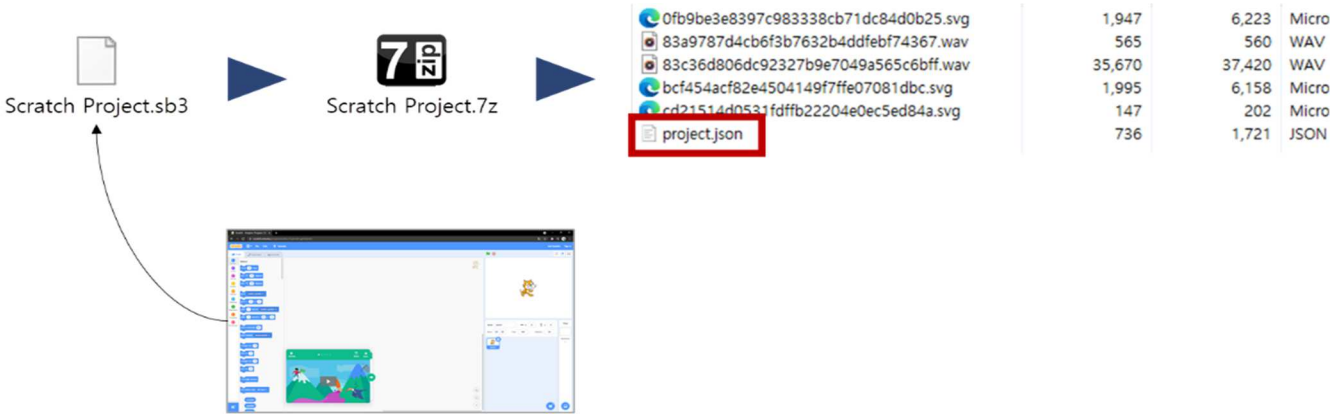

Fig. 1 The process of opening the files that make up the project

Changing the file extension to 7Z and then unzipping it displays the list of files contained in the project. The sprite and background image files used in the project could be downloaded in SVG format, and the sound source could be used in WAV format. In addition, there is one JSON-type file, which can be seen as a file containing the relationship between the code and the object.

Fig. 2 The block code of the project is an example.

It is assumed that a scratch project of Fig. 3 consists of one block code, as shown in Fig. 2. When checking the JSON file through Fig. 1, it can be confirmed that this project file consists of the code as shown in Fig. 4.

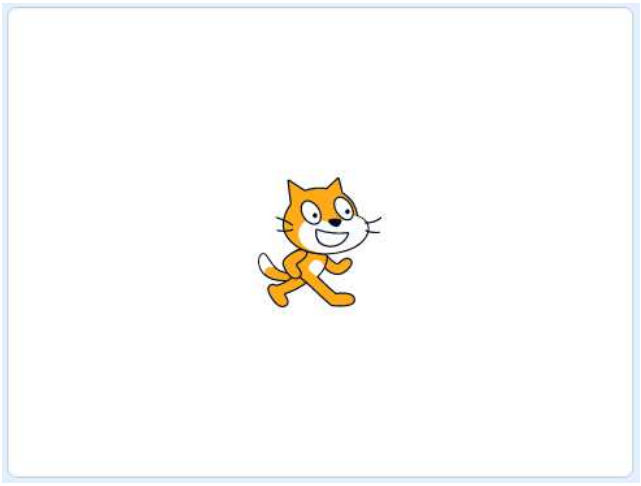

Fig. 3 Sprite screen for example project


```
{"targets":[{"isStage":true,"name":"Stage","variables":{"`jEk@4|i[#Fk?(8x)AV.-my variable":
["my variable",0]},"lists":{},"broadcasts":{},"blocks":{},"comments":{},"currentCostume":0,
"costumes":[{"assetId":"cd21514d0531fdffb22204e0ec5ed84a","name":"backdrop1",
"md5ext":"cd21514d0531fdffb22204e0ec5ed84a.svg","dataFormat":"svg","rotationCenterX":240,
"rotationCenterY":180}],"sounds":[{"assetId":"83a9787d4cb6f3b7632b4ddfebf74367","name":"pop",
"dataFormat":"wav","format":"","rate":48000,"sampleCount":1123,
"md5ext":"83a9787d4cb6f3b7632b4ddfebf74367.wav"}],"volume":100,"layerOrder":0,"tempo":60,
"videoTransparency":50,"videoState":"on","textToSpeechLanguage":null},{"isStage":false,
"name":"Sprite1","variables":{},"lists":{},"broadcasts":{},"blocks":{"``(DOq,pfy{v`YKhXDN`M":
{"opcode":"event_whenflagclicked","next":null,"parent":null,"inputs":{},"fields":{},
"shadow":false,"topLevel":true,"x":410,"y":379}},"comments":{},"currentCostume":0,"costumes":[
{"assetId":"bcf454acf82e4504149f7ffe07081dbc","name":"costume1","bitmapResolution":1,
"md5ext":"bcf454acf82e4504149f7ffe07081dbc.svg","dataFormat":"svg","rotationCenterX":48,
"rotationCenterY":50},{"assetId":"0fb9be3e8397c983338cb71dc84d0b25","name":"costume2",
"bitmapResolution":1,"md5ext":"0fb9be3e8397c983338cb71dc84d0b25.svg","dataFormat":"svg",
"rotationCenterX":46,"rotationCenterY":53}],"sounds":[
{"assetId":"83c36d806dc92327b9e7049a565c6bff","name":"Meow","dataFormat":"wav","format":"",
"rate":48000,"sampleCount":40681,"md5ext":"83c36d806dc92327b9e7049a565c6bff.wav"}],
"volume":100,"layerOrder":1,"visible":true,"x":0,"y":0,"size":100,"direction":90,
"draggable":false,"rotationStyle":"all around"}],"monitors":[],"extensions":[],"meta":
{"semver":"3.0.0","vm":"0.2.0-prerelease.20220322143111","agent":"Mozilla/5.0 (Windows NT 10.
0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/99.0.4844.84 Safari/537.36"}}
```


Fig. 4 JSON scripts for example projects

B. Data Structure and Definition of Action Type

The purpose of this study is to derive the system's requirements to be used by introductory programming learners. It is necessary to know which system interfaces interact for the requirements to be derived in this study. The system flow defined in this study was shown in Fig. 5 in the form of learners manipulate the client, and the data is processed in the server and stored in the database.

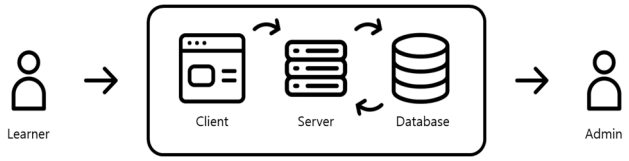

Fig. 5 System Configuration Interface

An interface requirement analysis method, one of the methods for deriving requirements, is an analysis method for classifying and specifying functional and non-functional requirements in a requirement specification [24]. As shown in Fig. 5, the system flow confirmed that the data generated through the learner's input was as shown in Fig. 4. Through this data, it is necessary to define what outcome the learner's data will be represented as from the system administrator's perspective [25].

First, the data structure was examined for parsing of Raw Data. The system to derive the requirements in this study compares the data generated when learners generate programming events to extract what the changed code is. The user can identify what behavior the learner has done through this extracted code. For this process to be possible, it is necessary to know the structure of the code in Fig.4. As a result of the researchers' analysis, the code of Fig. 4 had the same structure as Fig. 6 [26].

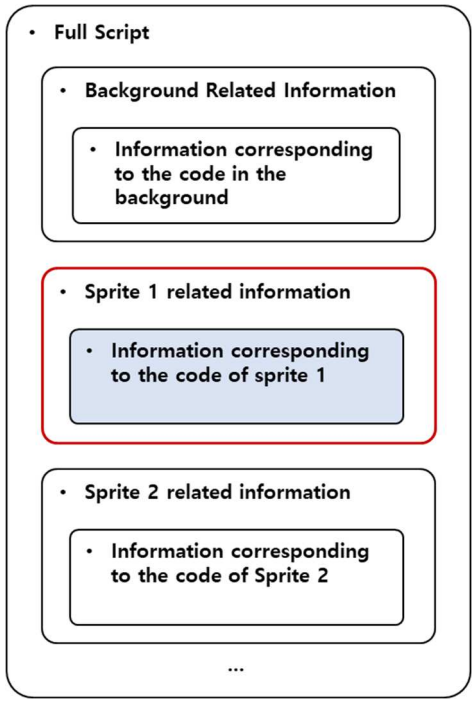

Fig. 6 JSON Script Configuration Diagram

If Fig. 6 is the code of Fig. 4, the center area highlighted with a red border is the script area corresponding to sprite 1 (cat) in Fig. 3. Since the code of Sprite 1 is edited when the Scratch editor is opened for the first time, the area corresponding to the colored rectangle is the code in Fig. 2. If there is a change in the code in this colored square area, the JSON script will change, and the system will also be able to recognize this change through parsing of the script.

Second, now that we understand the form of data, we need to define how to use the data. The data generated by the

learner's behavior should be defined in which behavior type to classify. In Scratch editing, there are many actions that learners can take. Among them, the types of actions that can occur in editing the block code are limited. Action 1, adding a block; action 2, deleting a block; action 3, copying the block; action 4, moving the block; action 5, changing the order of the block; action 6, changing the value inside the block. It consists of the action to do and action 7 to do nothing. In addition, if unexpected cases that require exception handling are considered, the number of possible cases is divided into a total of 8 types.

According to the intention of the programming log collection system, if expanding and checking the object editing case as well as the programming log, the learner's behavior type can be defined differently. As the Scratch programming process can be seen from the Scratch website community, many sprite users complete the project by the object editing process. We spend more than just programming time editing projects. Apart from code editing, 8 and 4 types of actions can occur when editing sprites or sound source files, respectively.

The types of actions that can occur in the sprite editing function are: First, the action to change the image format; second, the action to change the name of the sprite; third, the action to change the direction of the sprite; fourth, the action to change the size of the sprite, fifth, It consists of the action of changing the rotation direction of the sprite, sixth, editing the image file, seventh, adding the image file to an object, and eighth, deleting the image in one object.

In the case of the sound file editing function, many cases can be recognized on the system, but most are treated as unused or overlapping functions, and the number is merged into 4 cases. Editing behavior types of sound source files consist of first, changing the name of the sound source file, second, editing the sound source itself, third, adding the sound source file, and fourth, deleting the sound source file.

Depending on the scale of the system and the purpose of research or education, the data collected is sufficiently editable. The requirements to be derived in this study suggest all types of programming behaviors that learners can create. Through 20 requirements, it is suggested that the system should have, and its configuration and data type are defined.

III. Result and Discussion

Programming log analysis dependent on memory or experience can distort the results of the subjectivity of the researcher or learner. In order to collect objective programming logs, a system for collecting data must be developed. The system from which this study seeks to derive the requirements has constructed three interfaces for data collection. The requirements for displaying data generated by learners in seven behavioral types were derived from three interfaces: client, server, and database. Each requirement was derived based on the behavioral type to check the learner-generated data on the administrator page visually. The functional requirements required actually to operate, and the data types required to generate the data were derived together.

First, a scratch editor for the learner to perform scratch programming must be located in the case of a client. And at the same time, when multiple learners generate logs, an identifier is required to distinguish the learner's logs. Therefore, the client requires an "input" value, as shown in Table 1.

TABLE I
WEB FUNCTION REQUIREMENTS (FUC)

Target	Division	Requirements	data type	interface
Input	Input Name	The user can enter a name.	varchar2 (100)	Input Field
	Input Project Name	The user can enter a project name.	varchar2 (100)	Input Field
	Input Question	There is a button the user will press when they have a question.		Button
		There is space for the user to enter any questions they have.	varchar2 (400)	Input Field
	Input Submit	There is a button the user will press to submit the project.		Button
Alert	Check Question	A message is displayed to confirm that the question has been submitted.		Msg
	Check Submit	A message is displayed to confirm the submission.		Msg
Editor	Scratch Editor	It comes with the same editor as Scratch.		

The input value to be treated as an identifier is a minimum line that does not interfere with learning, and a total of two are included, the name of the learner and the name of the project. In addition, if this system was developed and research was conducted, two buttons that would serve as a sign in the learner's programming process were required as a convenience function. There may be a question button to register the question content when the learner does not know something during learning and a submission button to identify the status of the final code after the learner has finished coding until the end.

Second, the data transmitted by the client must be parsed through the server's processing phrase, transmitted to the database, and stored. The processing source code that must be on the server should include algorithms that can analyze behavior types and play a role in delivering them so that they can be stored in a database. Table 2 is a data classification table performed by a server.

TABLE II
DB FUNCTION REQUIREMENTS (FUC)

Target	Division	Requirements	data type	domain
Block Code	Add Code	It can be identified when a new code is added.	char(1)	"T", "F"
		You can see what type of code was added.	varchar2(32)	"OPCODE"
	Delete Code	It can be identified when two or more codes are copied.	char(1)	"T", "F"
		It can be identified when one or more codes have been erased.	char(1)	"T", "F"
	Move Code	It can be identified when the physical location (coordinate value) of the code is changed.	char(1)	"T", "F"
		You can see where the changed code is located.		{"x","y"}
		It can be identified when the relative position of the code (the order in which they are linked) has changed.	char(1)	"T", "F"
		It can be identified when chunks of code are combined.	char(1)	"T", "F"
	Modify Code	It can be identified when the value has changed.	char(1)	"T", "F"
		You can see what the changed value is.		
		It can be identified when there is no change.	char(1)	"T", "F"
	Copy the Code	It can be identified when the code is duplicated as the sprite is duplicated.	char(1)	"T", "F"
Sprites	Edit Object Properties	It can be identified when the image format is changed.	char(1)	"T", "F"
		It can be identified when the name is changed.	char(1)	"T", "F"
	Edit Object Information	It is possible to identify when a sprite's direction has changed.	char(1)	"T", "F"
		It can be identified when the size of the sprite is changed. (In Property)	char(1)	"T", "F"
		It is possible to identify the direction in which the sprite rotates. (In Block Code)	char(1)	"T", "F"
	Edit Object Image	It is possible to identify whether an image file has been edited.	char(1)	"T", "F"
		It is possible to identify whether an image object has been added. (Costume)	char(1)	"T", "F"
		It is possible to identify whether an image object has been deleted.	char(1)	"T", "F"
Sounds	Edit Object Information	It is possible to identify whether the name of the sound file has been changed.	char(1)	"T", "F"
	Edit Object Sound	It is possible to identify whether or not the sound file has been edited.	char(1)	"T", "F"
		It is possible to identify whether a sound file was added or not.	char(1)	"T", "F"
		It is possible to identify whether a sound file has been deleted or not.	char(1)	"T", "F"

It is difficult to define requirements while the type of data stored in the database is not defined. This study prepared a specification assuming an algorithm that must actually work to summarize the functional requirements for implementation as a result. Therefore, most of the required results consisted of requirements to determine the form of truth and falsehood and whether they worked. For example, results such as "true" and "false" are given to the requirement that "new code can be written."

There are cases where it should be treated as an exception here. It is difficult for the system administrator to determine whether the code has been copied or whether one code has been added. If receiving data directly from the client, the event can be checked, but considering the number of cases, the volume of data increases if receiving mouse events. Therefore, adding the number of cases to the algorithm is recommended so that the case copied through the script result can be confirmed. When there are two or more generated codes, it may be recognized that the codes are copied, and copying one or fewer codes was treated as adding codes. In addition, we derive a total of 12 requirements, including the combination of codes, the full copy of sprites, and the receipt of what kind of code was actually copied, to derive a requirement specification from obtaining all of the programming logs.

IV. CONCLUSION

This study analyzed the programming log of scratches to identify areas of data that can be extracted and derived the number of cases in which learners' output as intended. Functional requirements were derived by analyzing the structure of the data and the expected interface rather than the developed system. The minimum convenience of programming learners and managers or researchers analyzing logs was considered, and all functions necessary for the collection were included.

The system to be developed by satisfying this requirement aims to analyze programming learners' programming difficulties and assist teachers in education. In order to achieve this purpose, the person analyzing the learning process must understand the learner's programming process well. However, if one teacher has more than ten students, it is practically difficult to analyze the programming log of all learners. A system should be accompanied to objectively analyze the learner's log and give feedback to the learner based on the log. A system that reflects the requirements derived through this study will be able to generate important

clues for analyzing learners' behavior patterns. It is expected that the output of this study can be a key clue to analyzing learners' programming behavior.

ACKNOWLEDGMENT

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korean government (MSIT) (No. NRF-2021R1A2C2013735).

REFERENCES

[1] Lawan, Abdulmalik Ahmad, et al. "What is Difficult in Learning Programming Language Based on Problem-Solving Skills?." 2019 International Conference on Advanced Science and Engineering (ICOASE). IEEE, 2019.

[2] Thames, Lane, and Dirk Schaefer. "Software-defined cloud manufacturing for industry 4.0." Procedia cirp 52 (2016): 12-17.

[3] Lasi, Heiner, et al. "Industry 4.0." Business & information systems engineering 6.4 (2014): 239-242.

[4] Williamson, Ben. "Silicon startup schools: Technocracy, algorithmic imaginaries and venture philanthropy in corporate education reform." Critical studies in education 59.2 (2018): 218-236.

[5] Bosse, Yorah, and Marco Aurélio Gerosa. "Why is programming so difficult to learn? Patterns of Difficulties Related to Programming Learning Mid-Stage." ACM SIGSOFT Software Engineering Notes 41.6 (2017): 1-6.

[6] Pratiwi, Merina. "Student Tutoring, Facilitator and Explaining Models: A Problem Solving Metacognition towards Learning Achievements of Informatics Students." Journal of Educational Sciences 4.2 (2020): 368-379.

[7] Psycharis, Sarantos, and Maria Kallia. "The effects of computer programming on high school students' reasoning skills and mathematical self-efficacy and problem solving." Instructional science 45.5 (2017): 583-602.

[8] Hsu, Ting-Chia, Shao-Chen Chang, and Yu-Ting Hung. "How to learn and how to teach computational thinking: Suggestions based on a review of the literature." Computers & Education 126 (2018): 296-310.

[9] Shute, Valerie J., Chen Sun, and Jodi Asbell-Clarke. "Demystifying computational thinking." Educational Research Review 22 (2017): 142-158.

[10] Price, Thomas W., Yihuan Dong, and Dragan Lipovac. "iSnap: towards intelligent tutoring in novice programming environments." Proceedings of the 2017 ACM SIGCSE Technical Symposium on computer science education. 2017.

[11] Bhaugeerutty, Vinod Sharma. "Difficulties in Learning and Teaching Programming at Lower Secondary Level in Mauritius." Journal of Contemporary Research in Social Sciences 3.3 (2021): 48-61.

[12] Bosse, Yorah, and Marco Aurelio Gerosa. "Difficulties of programming learning from the point of view of students and instructors." IEEE Latin America Transactions 15.11 (2017): 2191-2199.

[13] Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., ... & Kafai, Y. (2009). Scratch: programming for all. Communications of the ACM, 52(11), 60-67.

[14] Weintrop, David, et al. "Starting from Scratch: Outcomes of early computer science learning experiences and implications for what comes next." Proceedings of the 2018 ACM conference on international computing education research. 2018.

[15] Microsoft MakeCode for micro:bit", Microsoft MakeCode for micro:bit, 2022. [Online]. Available: https://makecode.microbit.org/. [Accessed: 03- Jan- 2022]

[16] Kalelioglu, Filiz, and Sue Sentance. "Teaching with physical computing in school: the case of the micro: bit." Education and Information Technologies 25.4 (2020): 2577-2603.

[17] Bennedsen, Jens, and Michael E. Caspersen. "Revealing the programming process." Proceedings of the 36th SIGCSE technical symposium on Computer science education. 2005.

[18] Lye, S. Y., & Koh, J. H. L. (2018). Case studies of elementary children's engagement in computational thinking through scratch programming. In Computational thinking in the STEM disciplines (pp. 227-251). Springer, Cham.

[19] Lye, Sze Yee, and Joyce Hwee Ling Koh. "Case studies of elementary children's engagement in computational thinking through scratch programming." Computational thinking in the STEM disciplines. Springer, Cham, 2018. 227-251.

[20] Qarbon. Accessed 2009. Viewlet Builder. http://www.qarbon.com/

[21] Boisvert, Charles R. "A visualisation tool for the programming process." Proceedings of the 14th annual ACM SIGCSE conference on Innovation and technology in computer science education. 2009.

[22] Černý, Michal. "Measurement of selected aspects of student behavior in online courses." E-Pedagogium 19.2 (2019).

[23] Chang, Zhong, et al. "Scratch analysis Tool (SAT): a modern scratch project analysis tool based on ANTLR to assess computational thinking skills." 2018 14th International Wireless Communications & Mobile Computing Conference (IWCMC). IEEE, 2018.

[24] Lutz, Robyn R. "Targeting safety-related errors during software requirements analysis." Journal of Systems and Software 34.3 (1996): 223-230.

[25] Fitzgerald, Alissa M., Carolyn D. White, and Charles C. Chung. "What Is the Product? Requirements Analysis." MEMS Product Development. Springer, Cham, 2021. 31-45.https://en.scratch-wiki.info/wiki/JSON