KINETIK, Vol. 2, No. 4, November 2017, Pp. 327-336
ISSN : 2503-2259
E-ISSN : 2503-2267

The Use of Python Tutor on Programming Laboratory Session: Student Perspectives

Oscar Karnalim*[1], Mewati Ayub[2]
[1,2]Universitas Kristen Maranatha
oscar.karnalim@it.maranatha.edu*[1], mewati.ayub@it.maranatha.edu[2]

Abstract

Based on the fact that the impact of educational tools can only be accurately measured through student-centered evaluation, this paper proposes a long-term in-class evaluation for Python Tutor, a program visualization tool developed by Guo. The evaluation involves 53 students from 4 Basic Data Structure classes, which were held in the even semester of 2016/2017 academic year. It is conducted based on questionnaire survey asked to the students after they have used Python Tutor in their half of programming laboratory sessions. In general, there are three findings from this work. Firstly, Python Tutor helps students to complete programming laboratory tasks, specifically for Basic Data Structure material. Secondly, Python Tutor helps students to understand general programming aspects which are execution flow, variable content change, method invocation sequence, object reference, syntax error, and logic error. Finally, based on student perspectives, Python Tutor is a helpful tool positively affecting the students.

Keywords: *Questionnaire Survey, Program Visualization, Educational Tool, Laboratory Session, Programming*

1. Introduction

According to the fact that learning programming is a non-trivial task for students [1]–[4], especially for the novice ones, numerous educational tools have been developed to simplify such task. These tools aim to help at many levels, starting from algorithm to implementation level. In most occasions, they are developed with visualization as its main concern since such feature is believed being capable of enhancing student understanding further. A visualization-centric educational tool aiming to algorithm as its target material is frequently referred as Algorithm Visualization (AV) tool [5] whereas such tool aiming to programming (i.e. implementation level) as its target is referred as Program Visualization (PV) tool [6].

Python Tutor is a PV tool designed as a web-based application with responsive UI [7]. It can be accessed from anywhere and can be used on various machines such as personal computers, laptops, tablets, or smartphones. In this paper, the use of such tool in Data Structure laboratory session will be evaluated through a long-term in-class evaluation. In general, there are three objectives aimed in this work which are: 1) evaluating Python Tutor's impact for completing programming laboratory task per data structure material; 2) evaluating Python Tutor's impact for understanding programming aspects during programming laboratory session; and 3) collecting student experiences about the use of Python Tutor in programming laboratory session. The results of these objectives will be collected through questionnaire survey.

The survey was given to 53 students from 4 Basic Data Structure classes, which were held in the even semester of 2016/2017 academic year. To mitigate the bias, before asked to complete the survey, the students should use Python Tutor on half of the course sessions. They were required to use the tool for completing their programming laboratory task, particularly in understanding their own code and errors. The findings of this work are expected to provide a brief insight for Information Technology (IT) lecturers who plan to incorporate Python Tutor as their supplementary learning tool. They could exploit the positive impacts of Python Tutor reported in this paper while mitigating the negative ones.

2. Related Works

In order to tackle the emerging issues regarding to university students, several student-centered researches have been developed. Some examples of such researches are source code plagiarism detection [8], alumni tracer [9], student outcome prediction [10], and educational tool

Paper submitted on August 17, 2017; Revision on October 27, 2017; Received October 30, 2017

development [11]. These researches are often referred as student-centered researches since they heavily rely on student data, such as student experience, behavior, and achievement, during their development and/or evaluation. Yet, when compared to other researches, educational tool development is the most student-centered one since its effectiveness can only be measured based on student perspectives and grades.

In IT major, educational tools are often developed to teach non-trivial materials such as algorithm and programming. On the one hand, for teaching algorithm, educational tools are roughly classified into two major categories which are conventional and Algorithm Visualization (AV) tools. Conventional tools refer to educational tools developed as a standard GUI application. Complexitor [12], [13], aming to teach algorithm complexity in empirical manner, is an example which falls into this category. Algorithm Visualization (AV) tools; however, refer to educational tools focused on visualization as its main components [5]. These tools aim to teach how standard algorithms work through descriptive visualization. VisuAlgo [14], [15], AP-ASD1 [16], AP-SA [17], and AP-BB [18] are several tools falling into this category. On the other hand, for teaching programming, most educational tools are focused on visualizing and animating program aspects based on its runtime execution [6]. Several examples of such kind of tools are Jeliot 3 [19], JIVE [20], VILLE [21], and Python Tutor [7]. Among these mentioned tools, Python Tutor is the only tool which is designed as a web-based application.

Python Tutor is a web-based program visualization tool which is initially focused on visualizing Python programming language [7]. However, as the further development of Python Tutor is conducted, several popular programming languages such as Java and C++ are also incorporated. Unlike other program visualization tools, Python Tutor is designed as a web-based application with responsive UI for the sake of accessibility and simplicity of use. Users can access it from anywhere as long as they are connected with the Internet. Moreover, they can use it on various machines, either on personal computers, laptops, tablets, or smartphones.

According to the fact that several educational tools have been evaluated on real programming courses to comprehensively measure their effectiveness [4], [11], this paper proposes a long-term in-class evaluation about the use of Python Tutor in programming laboratory session. To the best of our knowledge, there is no related work discussing such topic. For our case study, we use 4 classes of Basic Data Structure course which were conducted on even semester of 2016/2017 academic year. The students were required to use such tool for completing laboratory task in half of the course sessions (7 of 14 laboratory sessions) and were asked to fill up a questionnaire at the end of the course. The questionnaire results were then reported as the results of this paper.

3. Research Method

In general, there are three objectives that will be measured in this paper. These objectives are: 1) evaluating Python Tutor's impact for completing programming laboratory task per data structure material; 2) evaluating Python Tutor's impact for understanding programming aspects during programming laboratory session; and 3) collecting student experiences about the use of Python Tutor in programming laboratory session. These objectives will be achieved by collecting the respondent's answers according to the questionnaire survey. Our respondents are undergraduate students from 4 Basic Data Structure classes in even semester of 2016/2017 academic year, where most of the students are from the class of 2016. The detail of respondent statistics toward these classes can be seen in Table 1. For each class, its total number of respondents was lower than its total number of students since some students might drop the class at the middle of the semester or did not come at the questionnaire session. Despite such issues, in general, the proportion of involved respondents toward class students is still considerably high (85.483%).

To mitigate questionnaire biases, before the respondents were asked to complete the questionnaire, they should experience two kinds of programming laboratory session for one semester. One of them was the session that was intervened with Python Tutor whereas the other one was the conventional one (without the use of Python Tutor). Both kinds of session will be conducted on 4 classes alternately where the session distribution details can be seen on Table 2. In general, Python Tutor's intervention will be applied at odd weeks for class C and D and even weeks for class A and B. We intentionally put the intervened session alternately among classes so that we can gather the impact of Python Tutor for all course materials while providing conventional laboratory session as a baseline for the respondents. Each time a session was intervened with the use of Python Tutor, the respondents will be asked to understand their own code and errors through the information provided on Python Tutor. In other words, we encouraged the respondents to use Python Tutor as a supplementary aid for completing the programming laboratory tasks.

Table 1. Respondent Statistics

Class	Total Number of Students	Total Number of Respondents
A	15	14
B	10	9
C	19	14
D	18	16
Total	62	53

Table 2. The Intervention Schedule of Python Tutor

Week	Class			
	A	B	C	D
1st week: The Introduction of Abstract Data Type (ADT)			✓	✓
2nd week: Simple Interaction between ADTs	✓	✓		
3rd week: Array of ADT			✓	✓
4th week: ADT Array	✓	✓		
5th week: ADT Stack			✓	✓
6th week: ADT Queue	✓	✓		
7th week: Insertion and Deletion of ADT Linked List			✓	✓
8th week: Supplementary Methods of ADT Linked List	✓	✓		
9th week: Interaction between ADT Linked List			✓	✓
10th week: ADT Queue with Linked List as Its Internal Structure	✓	✓		
11th week: ADT Priority Queue			✓	✓
12th week: ADT Double-pointer Linked List	✓	✓		
13th week: ADT Circular Linked List			✓	✓
14th week: Shell and Merge Sort	✓	✓		

At the end of week 14th, all respondents were asked to fill a questionnaire where its questions reflected our three objectives. It consisted of 14 questions which details can be seen on Table 3. For convenient reference at the rest of this paper, each question was assigned with a unique ID. Generally speaking, the questions were classified into three categories regarding to its objective. Q1-Q7 referred to the first objective; Q8-Q13 referred to the second objective; and Q14 referred to the last one. It is important to note that the intervened sessions referred by Q1-Q7 relied heavily on the student class. For example, the 1st intervened session of class A and B was the 2nd course week discussing about simple interaction between ADTs. It was different with the 1st intervened session of class C and D, which was the 1st course week discussing about the introduction of ADT. These differences were the consequences of our proposed session distribution where not all classes were intervened with Python Tutor on similar course material.

Q1-Q7 should be answered in 7-points Likert scale where 1 represented extremely strong disagreement, 2 represented strong disagreement, 3 represented weak disagreement, 4 represented neutral, 5 represented weak agreement, 6 represented strong agreement, and 7 represented extremely strong agreement. Respondent answers of these questions were used to evaluate which course material was affected the most and the least by Python Tutor. Q8-Q13 should be answered in similar manner with Q1-Q7. They should be answered in 7-points Likert scale. Respondent answers of these question were used to evaluate which programming aspect was affected the most and the least by Python Tutor. In our case, we incorporated 6 programming aspects which were execution flow, variable content change, method invocation sequence, object reference, syntax error, and logic error. These aspects were involved on questionnaire questions from Q8 to Q13 respectively. Q14 was an open question, meaning that it should be answered with several natural language sentences. This question aimed to provide compiled student experiences regarding to Python Tutor's usage in programming laboratory session.

Table 3. Survey Questions

ID	Statement
Q1	Python Tutor helps student to complete programming laboratory task on the 1st intervened session
Q2	Python Tutor helps student to complete programming laboratory task on the 2rd intervened session
Q3	Python Tutor helps student to complete programming laboratory task on the 3rd intervened session
Q4	Python Tutor helps student to complete programming laboratory task on the 4th intervened session
Q5	Python Tutor helps student to complete programming laboratory task on the 5th intervened session
Q6	Python Tutor helps student to complete programming laboratory task on the 6th intervened session
Q7	Python Tutor helps student to complete programming laboratory task on the 7th intervened session
Q8	Python Tutor helps student to understand the execution flow of running program
Q9	Python Tutor helps student to understand how variable content changes
Q10	Python Tutor helps student to understand the sequence of method invocation
Q11	Python Tutor helps student to understand the concept of object reference
Q12	Python Tutor helps student to understand syntax error
Q13	Python Tutor helps student to understand logic error
Q14	Please provide one of your own experiences about the use of Python Tutor in programming laboratory session.

4. Results and Discussion

4.1 The Result of Evaluating Python Tutor's Impact in Laboratory Session per Course Material

The results of questionnaire survey regarding Python Tutor's impact in laboratory session per data structure material (Q1-Q7) was split into twofold which were the results of the odd weeks (from class C and D) and the results of the even weeks (from class A and B). These results, which are displayed as box-and-whisker plots, can be seen on Figure 1 and Figure 2 respectively. Generally speaking, all intervened sessions yield positive feedbacks since average values for each statement (displayed as an *x* symbol on the plot) are higher than 5 (more than weak agreement). Some boxes from the result of the odd weeks are smaller than the results of the even weeks since the respondents from class C and D tend to share higher agreement level when compared to the respondents from the remaining classes.

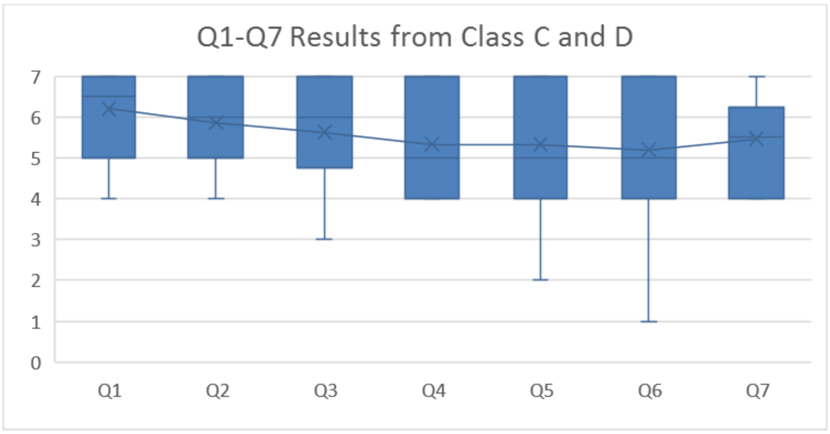

Figure 1. Q1-Q7 The Results for Odd Weeks, Collected from Class C and D

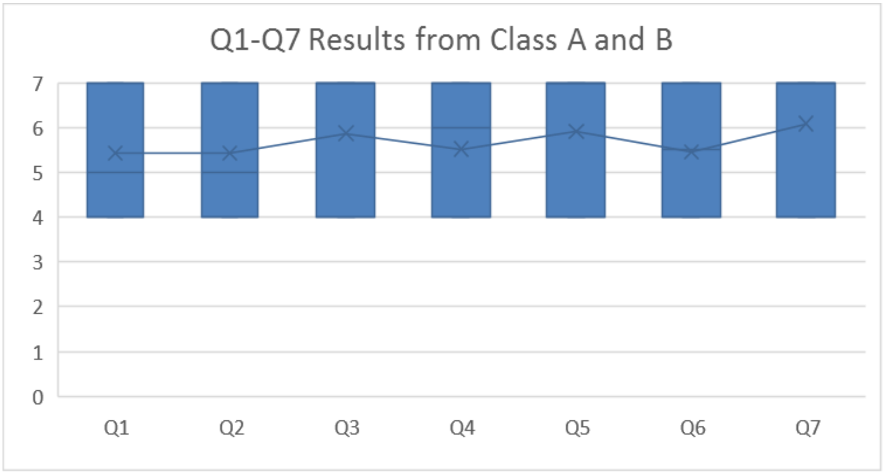

Figure 2. Q1-Q7 The Results for Even Weeks, Collected from Class A and B

According to Figure 1, for class C and D, the 1st intervened session yields the highest average score whereas the 6th intervened session yields the lowest average one. On the one hand, these respondents agree that Python Tutor helps students the most when completing laboratory task about the introduction of ADT. Such finding is natural since the task given on such session was the simplest one when compared to other tasks. Simple task may enhance respondent's focus toward the information provided on Python Tutor since their focus will not be distracted by the task difficulty. There are two reasons why the task given on such session is the simplest one: 1) the session was given at the beginning of the course, having the lowest task difficulty. As we know, in most occasions, the materials given on a course would have a non-decreasing difficulty, started from the lowest one at the beginning of the course; and 2) in our case, the task given on that session was split into several independent smaller sub-tasks. Therefore, it might be easier to complete the given tasks since the scope for each sub-task should be narrower than the task itself due to their problem independence. On the other hand, these respondents agree that Python Tutor helps students the least when completing laboratory task about ADT Priority Queue. Even though its difficulty is quite similar with Circular Linked List on the 7th intervened session, the respondents feel such task is still more difficult since they were asked to write *enqueue* method from ADT Priority Queue. Such method involves three insertion mechanisms which might confuse the respondents when completing the task.

Several results from Figure 1 have a long whisker either at the top or the bottom of the box. It means that several respondents have uncommon response. For Q1, Q2, Q3, Q5, and Q6, several respondents provide lower response than the usual one since they prefer conventional laboratory session rather than the intervened one. For Q7, only several respondents put 7 as their responses since, according to our informal in-class observation, they feel that Python Tutor's reference visualization for Circular Linked List is not descriptive enough. Python Tutor put several edges to visualize object references. Yet, such edges become visually complicated when the pointer of Circular Linked List's last element refers to its first element.

According to Figure 2, for class A and B, the 7th intervened session yields the highest average score whereas the 1st and 2nd sessions yield the lowest average one. The 7th intervened session, which material is shell and merge sort, is considered as the most assisted session since it was the only task from class intervened sessions which was split into two independent sub-tasks. Splitting the task into smaller independent sub-tasks might generate simpler task, which might help the respondents to focus on the information provided on Python Tutor rather than the task difficulty. The 1st and 2nd intervened sessions, on the contrary, are considered as the least assisted sessions since some respondents in class A and B have never used the tool before, and they need to adapt themselves with the tool while completing the task. It is quite different with the respondents from class C and D where almost all of them have used the Python Tutor beforehand in Introductory Programming, a predecessor of our evaluated course. As seen in Figure 2, there is no whisker shown either at the top or the bottom of the boxes. Thus, it can be concluded that all respondents from class A and B have a high level of agreement regarding to their responses.

4.2 The Result of Evaluating Python Tutor's Impact in Laboratory Session per Programming Aspect

The result of questionnaire survey regarding Python Tutor's impact in laboratory session per programming aspect (Q8-Q13) can be seen on Figure 3 as a box-and-whisker plot. Q8-Q13 are designed to evaluate the impact of Python Tutor toward the execution flow, variable content change, method invocation sequence, object reference, syntax error, and logic error respectively. In general, all programming aspects are affected positively since average values for each statement (displayed as an *x* symbol on the plot) are higher than 5 (more than weak agreement).

Among the evaluated aspects, execution flow (Q8) yields the highest average value based on twofold. On the one hand, such aspect is inseparable to source code. The students cannot create any source code if they do not understand about execution flow. Other aspects such as object reference and method invocation do not generate such significant impact since they are only required on several occasions while creating the source code. The students can still create some codes even though they do not understand these aspects. On the other hand, such aspect is the main concern of Python Tutor as a PV tool. It is natural that the programming aspect included as the main concern becomes the most affected aspect by the use of that tool.

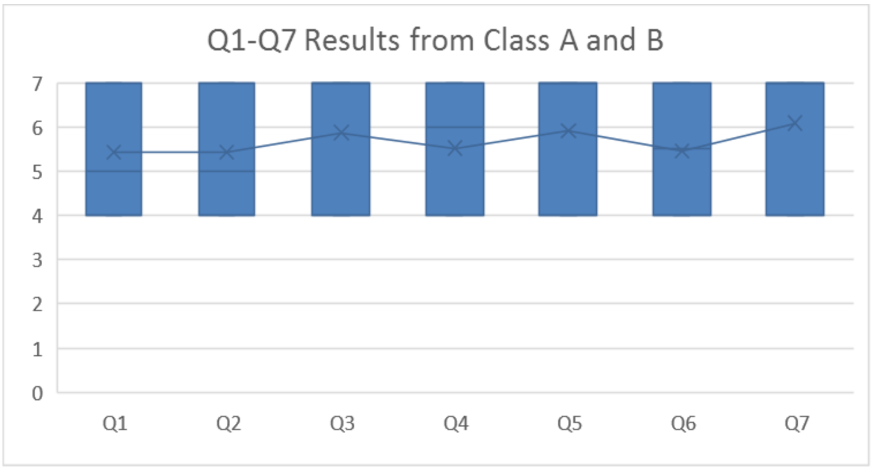

Figure 3. Q8-Q14 The Results from Four Classes, A, B, C, and D

Q13, which is focused on evaluating whether Python Tutor helps student to understand logic error, yields the lowest average score among other aspects. Based on our in-class observation, most respondents tend to check such error on standard IDE (i.e. Python IDLE) instead of Python Tutor. At first, they will use Python Tutor to check how their code works. However, if a logic error occurred, they will copy the code to standard IDE and track the error utilizing this interface. When informally asked about the reason of this alteration, the respondents answer that the standard IDE is more convenient for them to solve an error. Such behavior is also conducted by the respondents when the syntax error is occurred. That is why Q12, which is focused on evaluating whether Python Tutor helps student to understand syntax error, also yields a considerably low average score when compared to other statements.

Several results from Figure 3 have a long whisker at the bottom of the box since some respondents think that such tool is not beneficial for helping students to understand some programming aspects. Q11, which is focused on evaluating whether Python Tutor helps student to understand the concept of object reference, is the only statement which has no whisker at the bottom of the box. Thus, it can be stated that all respondents have high level agreement regarding such impact, even though its average value is considerably low when compared to other statements.

4.3 Compiled Student Experiences about Python Tutor Usage in Programming Laboratory Session

The compiled results of questionnaire survey regarding to student experiences about the use of Python Tutor in programming laboratory session (Q14) can be seen on Table 4. These experiences are generalized based on their main effect and displayed in decreasing order of occurrences. For convenient reference at the rest of this paper, each experience is assigned with a unique ID. It is important to note that such experiences are only taken from 48 of 53 respondents since the remaining 5 respondents did not provide detailed experiences. They only stated that such tool had helped them a lot. In addition, since we limit the respondents to write only one experience, it can be assumed that their responses would be the most memorable one for them.

Table 4. Compiled Student Experiences

ID	Experience	Occurrences
E1	Python Tutor helps the respondents to find errors	16
E2	Early adaptation to Python Tutor takes a considerable amount of time	12
E3	Python Tutor helps the respondents to know how their code work	8
E4	Slow Internet connection discourages the respondents to use Python Tutor	4
E5	Several technical issues are occurred while copying the code to Python Tutor	4
E6	Python Tutor does not help the respondents to solve error recognition	2
E7	Python Tutor helps the respondents to sharpen their logical thinking	1
E8	Python Tutor helps the respondents to complete their programming task faster	1

E1 experience, which is felt by 16 respondents, claims that Python Tutor helps the respondents to find errors. The respondents argued that several errors (either syntax or logic) could be found efficiently by visualizing the program through Python Tutor. However, even though such tool helps them to find the errors, they did not state that such tool was also valuable for solving the errors. From other respondent experiences (E6), 2 respondents even explicitly stated that such tool did not help them to solve the errors. When discovered further, these respondents feel that standard IDE have more comprehensive features for solving errors rather than Python Tutor. This finding is natural since Python Tutor is not specifically designed to help student for solving errors.

E2 experience, which is felt by 12 respondents, claims that Python Tutor is quite difficult to be used for the first time. The respondents stated that they took a considerable amount of time before getting used to it. According to their experiences, Python Tutor’s UI is less intuitive for them as the first-time users. However, we believe that this issue could be easily handled by providing proper and comprehensive tutorial session beforehand.

E3 experience, which is felt by 8 respondents, claims that Python Tutor helps the respondents to know how their code works. The respondents stated that Python Tutor had several unique features which helped them to understand more about their code’s running behavior. According to their experiences, among these features, source code line highlight and variable content view are the two most helpful ones. They stated that both features were descriptive enough to show code behavior, and they usually focus on these features to understand their code and error.

E4 experience, which is felt by 4 respondents, claims that slow Internet connection discourages the respondents to use Python Tutor. Slow connection slows down Python Tutor’s processes, particularly in user interaction. As a result, visualizing a program on such condition might take longer time than necessary. Such event is discouraging for most respondents since they should complete the programming task in a limited time. Lagged processes on Python Tutor might cut up their working time drastically. In most occasions, if the Internet connection is slow, the respondents do not use the Python Tutor and rely heavily on the standard IDE instead. In fact, this issue could be easily handled either by providing Python Tutor in an offline mode or increasing the bandwidth of the Internet connection. We will apply one of these solutions in the future use of Python Tutor.

E5, which is felt by 4 respondents, claims that several technical issues are occurred while copying the code to Python Tutor. Some of the respondents experience some issues regarding changed Python version whereas the others experience the issue regarding importing additional file/library. Similar with E2, we believe that such issues could be handled easily by providing a proper tutorial session.

E6, which is felt by 2 respondents, claims that Python Tutor do not help the respondents to solve found errors. The respondents stated that it was hard to track and solved found errors on Python Tutor. In contrast with standard IDE, Python Tutor is not featured with comprehensive features for tracking and solving errors. Thus, it is natural that the respondents would feel that way.

E7, which is felt by 1 respondent, claims that Python Tutor helps the respondent to sharpen his/her logical thinking. He/she believes that such tool has helped him/her a lot, especially for understanding execution logic on the program. E8, on the other hand, which is also felt by only 1 respondent, claims that Python Tutor helps the respondent to complete his/her programming task faster. He/she stated that, for his/her case, the tasks from intervened sessions were completed in shorter time when compared to tasks from conventional sessions.

5. Conclusions and Future Works

This paper presents the student perspectives toward the use of Python Tutor for completing data structure programming task in laboratory session. The perspectives were collected from 4 classes of Basic Data Structure course, which were held in even semester of 2016/2017 academic year. To avoid biased result, the student perspectives were only collected after the students had tried the tools for a half of the course sessions while experiencing the conventional laboratory session on the other half. According to our respondents, Python Tutor provides positive impacts for completing Basic Data Structure laboratory tasks and understanding general programming aspects (i.e. execution flow, variable content change, method invocation sequence, object reference, syntax error, and logic error). In addition, such tool also provides positive feedbacks when perceived from student experiences in general. Even though some of the experiences are the negative ones, we do believe that such positive impacts outweigh the negative ones, and several strategies can be applied to mitigate the negative effects.

For future works, we plan to evaluate the impact of Python Tutor through student's grade. Such results are expected to complement our current work so that we could see the impact from both perspectives: qualitative and quantitative perspective. Moreover, we also plan to develop an upgraded version of Python Tutor, which is focused to mitigate the negative feedbacks that are reported in this work. Hopefully, such tool may help students to learn programming, especially in our university.

References

[1] M. McCracken et al., "*A Multi-National, Multi-Institutional Study of Assessment of Programming Skill of First-year CS Students*," ACM SIGSCE Bulletin, Vol. 33, No. 4, 2001.

[2] R. Lister et al., "A *Multi-National Study of Reading and Tracing Skills in Novice Programmers*," ACM SIGSCE Bulletin, Vol. 36, No. 4, 2004.

[3] T. Chen et al., "*Students Designing Software: A Multi-National, Multi-Institutional Study*," Informatics Education, Vol. 4, No. 1, 2005.

[4] S. M. Cisar, R. Pinter, and D. Radosav, "*Effectiveness of Program Visualization in Learning Java: a Case Study with Jeliot 3*," International Journal of Computers, Communications & Control, Vol. 6, No. 4, 2011.

[5] J. Urquiza-Fuentes and J. Á. Velázquez-Iturbide, "*A Survey of Successful Evaluations of Program Visualization and Algorithm Animation Systems*," ACM Transactions on Computing Education (TOCE) - Special Issue on the 5th Program Visualization Workshop, Vol. 9, No. 2, 2009.

[6] S. Bentrad and D. Meslati, "*Visual Programming and Program Visualization- Toward an Ideal Visual Software Engineering System,*" ACEEE International Journal on Information Technology, Vol. 1, No. 3, 2011.

[7] P. J. Guo, "*Online Python Tutor: Embeddable Web-Based Program Visualization For CS Education*," in The 44th ACM technical symposium on Computer Science Education, 2013.

[8] O. Karnalim, "*Detecting Source Code Plagiarism on Introductory Programming Course Assignments Using a Bytecode Approach*," in The 10th International Conference on Information & Communication Technology and Systems (ICTS), 2016.

[9] H. Toba, E. A. Wijaya, M. C. Wijanto, and O. Karnalim, "*Enhanced Unsupervised Person Name Disambiguation to Support Alumni Tracer Study*," Global Journal of Engineering Education, Vol. 19, No. 1, 2017.

[10] M. Ayub and O. Karnalim, "P*redicting Outcomes in Introductory Programming Using J48 Classification*," World Transactions on Engineering and Technology Education (WTE&TE), Vol. 15, No. 2, 2017.
[11] E. Kaila, T. Rajala, M. J. Laakso, and T. Salakoski, "*Effects of Course-Long Use of a Program Visualization Tool*," in Australasian Computing Education Conference, 2010.
[12] E and O. Karnalim, "*Complexitor: An Educational Tool for Learning Algorithm Time Complexity in Practical Manner*," ComTech: Computer, Mathematics and Engineering Applications, Vol. 8, No. 1, 2017.
[13] O. Karnalim and Elvina, "*Interfacing Complexitor: An Empirical-based Educational Tool for Learning Time Complexity*," Journal of IRD (Informatics Research and Development), Vol. 1, No. 1, 2017.
[14] E. T. Y. Ling, "T*eaching Algorithms with Web-based Technologies*," Department of Computer Science, School of Computing, National University of Singapore, 2014.
[15] S. Halim, Z. C. Koh, V. B. H. Loh, and F. Halim, "*Learning Algorithms with Unified and Interactive Web-Based Visualization*," Olympiads in Informatics, Vol. 6, Pp. 53–68, 2012.
[16] L. Christiawan and O. Karnalim, "*AP-ASD1: An Indonesian Desktop-based Educational Tool for Basic Data Structures*," Information Technology and Information System, Vol. 2, No. 1, 2016.
[17] F. C. Jonathan, O. Karnalim, and M. Ayub, "*Extending The Effectiveness of Algorithm Visualization with Performance Comparison through Evaluation-integrated Development*," in National Seminar on Information tachnology Appication, 2016.
[18] S. Zumaytis and O. Karnalim, "*Introducing An Educational Tool for Learning Branch & Bound Strategy*," Journal of Information Systems Engineering and Business Intelligence, Vol. 3, No. 1, 2017.
[19] A. Moreno, N. Myller, E. Sutinen, and M. Ben-Ari, "*Visualizing programs with Jeliot 3*," in The Working Conference on Advanced Visual Interfaces, Galipoli, 2004.
[20] P. Gestwicki and B. Jayaraman, "*Interactive Visualization of Java Programs*," in Symposia on Human Centric Computing Languages and Environments, 2002.
[21] T. Rajala, M.-J. Laakso, M. Kaila, and T. Salakoski, "*VILLE - A Language Independent Program Visualization Tool*," in The 7th Baltic Sea Conference on Computing Education Research, 2007.