Embedded Self Organizing Systems (Vol 10. No 7. 2023) (pp.62-67)

Embedded Selforganizing Systems

Special Issue Topic: "International Symposium on Computer Science and Educational Technology"

Evaluation of Python Error Message Interpretation: Study on Students with Different Levels of Programming Experience

Gantogoo Oyunbat
Faculty of Computer Science
Chemnitz University of Technology
Chemnitz, Germany
gantogoo.oyunbat@informatik.tu-chemnitz.de

Ahmad Hamdy Sayed Hassanien
Faculty of Computer Science
Chemnitz University of Technology
Chemnitz, Germany
ahmad-hamdy-sayed.hassanien@informatik.tu-chemnitz.de

***Abstract*—Error messages are a helpful tool for everyone interested in learning a programming language. Even after learning the language, novice and experienced programmers alike, have to interact with an error message one way or another. However, there is no single programming language that is used across all platforms and systems, so the programmers have to write programs in many different languages. If the programming languages have similar structures, working with a new language is relatively easy. We have asked the question, whether the same effect could be observed in case of error messages. We designed an online survey which was conducted internationally to measure whether the general experience of programmers from any programming language influences the programmer's ability to correct python error message. The survey was aimed at students who have some experience with programming. We find in comparison with novices, on average the experienced programmers find and fix an error in the code given the same error message with its corresponding code snippet. Additionally, we see correlations between the average of correctly fixed errors and number of programming languages that the participants had experience with, as well as their chosen major and their age.**

***Index Terms*—programming experience, error message, python, student**

I. Introduction

A central part of learning any programming language is making mistakes and errors. When a mistake is made while programming, in case of the python programming language as well as many others, error messages are shown so that the programmer can fix (debug) their error (bug). However, more often than not the error messages are cryptic and hard to read. There are ongoing and previous researches related to error messages, their cause, readability and effectiveness in helping the programmer solve the problem [1]–[8]. We tried to concentrate our focus on the influence of the students' past experience in programming language, given error messages in python, on the likelihood of solving the error correctly and on the time necessary for solution.

It might seem trivial that the more experienced the programmer is it is more likely that they do better and faster at finding and correcting an error in the code, however it is not always the case as it was actually shown in the field of medicine that newly graduated doctors perform better than senior doctors because the medical knowledge is still fresh in their minds [9]. A similar case could be seen in mental health fields [10]. It is of interest to us to experiment whether the same could be also observed in case of programming.

This paper is organized as follows: Section II contains the research question, hypotheses, planning of experiment, participants, and material creation, Section III presents the collected data from the conducted survey, and the corresponding descriptive statistics, in Section IV we discuss our findings from the collected data and test its significance, additionally, possible threats to validity of the result are considered, lastly in Section V we conclude the topic and mention potential future works relating to the topic.

II. Study Design

A. Research Goals

Error messages should deliver the programmers, in particular to novice programmers, essential information regarding the location and cause of an error in the code [4,11,12]. Furthermore, error messages should avoid frustrating the programmer by being challenging to understand and giving useless information to the user [11]. These exact difficulties with error messages make programming seem hard and influence beginners negatively which sometimes make them reluctant in

learning a programming language [2]. Regarding this issue, Jadud mentioned in 2006 that error messages that commercial compilers generate are often misleading [13].

According to [4] error messages in modern programming languages are still cryptic, misleading and uninformative. It is true that even now, programmers have complications understanding an error message. Especially, if it is an error message from a programming language one is not acquainted with, the programmers have trouble finding the error and fixing the error.

B. Research Question

In this research the following research question are asked:

(RQ) How does the past programming experience of the student influence the likelihood of finding the corresponding mistake in the code, and the time to find it, given the python error message?

Here, the independent variables are the past experience of the student with programming in general, and the shown error message. The dependant variables are the likelihood of finding the mistake, and the time it takes to find it. The dependent and independent variables have been operationalized. As a clear distinction between experienced and inexperienced programmers is needed for the analysis, we considered students with equal to or more than a year's worth of experience in programming as more experienced and students with less than a year of experience as novices. Likelihood is measured by the amount of correctly found and fixed mistakes, per codes shown.

C. Hypotheses

With the operationalized variables we propose following hypotheses and their null hypotheses.

Hypotheses:

H1: Given the same python error message, experienced programmers are more likely to find and solve the corresponding mistake than inexperienced programmers.

H2: Given the same python error message, experienced programmers find the corresponding error faster than inexperienced programmers.

Null Hypotheses:

$\mathbf{H1}_0$: Given the same python error message, experience of the programmer has no effect on the likelihood in finding and solving the corresponding mistake.

$\mathbf{H2}_0$: Given the same python error message, experienced and inexperienced programmers will find the corresponding error in the code in the same amount of time.

We hypothesize that the experience of the students in a programming language influences their python error message interpretation positively, such as solving the existing error faster and more precisely. However, the python error message could also be sufficiently well written so that the non-experienced students can interpret the message just as correctly as their experienced colleagues.

D. Experiment Planning

The design of our experiment is intended to allow us to gain a comprehensive understanding of the relationship between past experience and the ability to interpret error messages in programming. By targeting students with programming experience, we aim to capture a broad range of perspectives and experiences, which will help us to build a more complete picture of the topic.

To reach a large and diverse participant pool, we have decided to conduct an online survey that can be accessed from anywhere in the world at any time of day. This approach allows us to gather data from a wider range of participants, which help us to draw more robust conclusions about the relationships between the level of past experience and error message interpretation.

As our experiment is focused on comparing the experiences of participants with different levels of programming experience, it can be seen as a between-group design experiment. This means that we will compare the results of participants who have different levels of experience, in order to identify the impact of experience on their ability to interpret error messages.

E. Participants

The participants are split into multiple groups, depending on their level of experience, age, major, gender, self assessment, and experience with number of programming languages. The level of experience is determined with various questions before the survey. These questions determine the following points:

- Whether the participant has studied or is studying any programming languages.
- Whether the participant is working or has worked in a programming related job.
- What was the first programming language the participant learned?
- When did the participant learn their first programming language?
- When was the last time the participant did programming?
- How confident are the participants in their program debugging abilities?

Using the answers to the questions above we classified our subjects into different groups of experience level. The questions also helped in determining data that could have unwanted effects on the result such as false data or outliers.

F. Material Creation

For our experiment we collected online responses from students around the world using the SoSci online platform. It was necessary to differentiate between questions that determine the programmer's level of experience, and questions that the ability to debugging the given errors. For this purpose, we divided the survey into two parts. Upon launching the survey, the participant is greeted with an introduction page that informs them how the survey will run, as well as the contact details of the researchers in case they have feedback. By clicking on next, the participant goes to the first part of

the survey. In the first part the participant is asked questions to help us classify their level of experience.

In the second part of the survey the participant is shown 10 error message examples and codes individually, and during the 10 minute time limit they would try to solve as much of these errors as possible. A time limit was introduced to later compare the speed of the participant for whole set of questions in regard to the correctness, rather than comparing time for individual questions. The survey ends, when the timer runs out or the participant finishes the survey.

III. Results

The data was collected for one month, beginning from December 20, 2022 until January 20, 2023. During this period 49 students took part in the survey. Upon closer inspection, 20 participants were excluded from further analysis due to missing time values in the second part of the survey, leaving 29 answers for the analysis.

A. Presentation of the Collected Data

After the exclusion of data from participants that did not reach the second part of the survey, the remaining data was split into several groups connecting the first and second part of the survey. For each group the average number of correctly answered questions in part two was calculated. With the calculation results graphs were drawn, with the average score on the y-axis and the related groups on the x-axis, to visualize and compare the data. During the visualization, it became apparent that the remaining data still had additional outliers which distorted the outcome of the graph. These outliers were participants who reached the second part of the survey but did not spend enough time on each question as well as the question set as a whole.

Another type of outliers were participants that qualify as inexperienced programmers according to the criteria, however they had very high scores. As we can see in Fig. 1, two outliers from the inexperienced group showed inconsistency within the group.

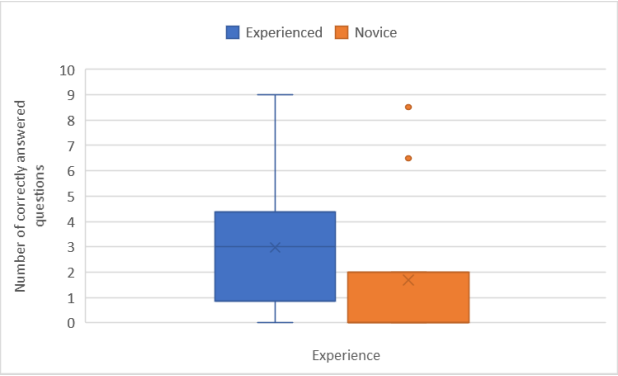

Fig. 1. Boxplot showing comparison between experienced and inexperienced programmers.

According to their answers, the two outliers are studying Computer Science in their fourth masters semester, that would mean that it is partly expected of them to have studied a programming language in their earlier semesters, therefore more than a year ago. By removing the outliers we are left with 20 participants.

Overall, a few patterns were seen: most participants were majoring in computer science, the rest were majoring in engineering. The subjects came from different countries such as Germany, Mongolia, Egypt, Iran, Pakistan, Mexico, Taiwan, and Kazakhstan. The majority of participants noted that they used their programming skills at work and knew several programming languages. The majority were also male.

B. Descriptive Statistics

The analysis of the average test scores involved a multi-faceted approach. Factors such as the age, gender, major, familiarity with programming languages, and previous experience with different programming languages were taken into consideration. This information was then visualized in Fig. 2. The aim of this analysis was to identify any correlations or trends between different variables and average test scores. By considering various perspectives, a comprehensive understanding of the data could be achieved.

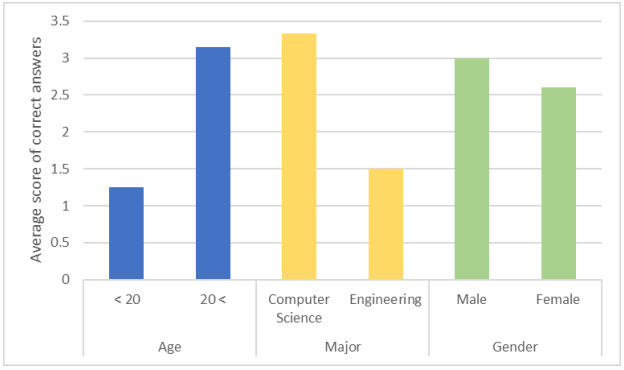

Fig. 2. Average score by age, major and gender.

The data analyzed suggest that age may play a role, as those who are older than 20 years old appear to have a higher success rate in fixing code errors. This may be due to the accumulation of experience and knowledge over time. Furthermore, the data indicate that individuals who have majored in computer science may be better equipped to solve code errors, as they have likely received a more in-depth education in programming and computer science principles. The survey results did not reveal any apparent differences in scores between male and female students. Additionally, the average score was graphed over the number of programming languages the participants had previous experience in, see Fig. 3.

The data presented in the Fig. 4 shows a correlation between proficiency in multiple programming languages, particularly including Python, and the ability to solve errors in code.

This suggests that having a wider range of programming knowledge can help with identifying and fixing mistakes in code. Additionally, the graphs show that those who use programming in their work are also more likely to have

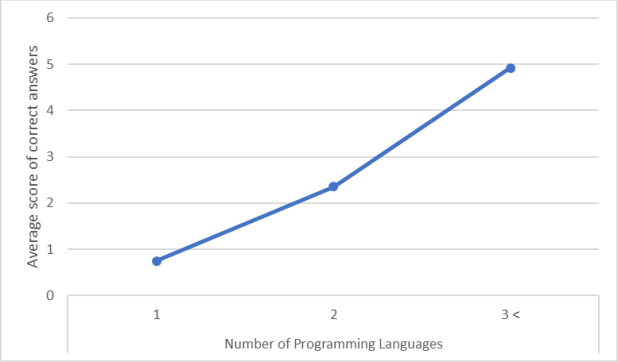

Fig. 3. Average score by number of programming languages.

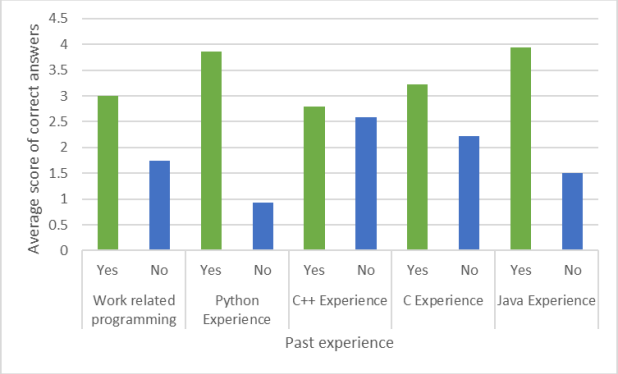

Fig. 4. Average score by past experience with different programming languages and work related programming.

success in fixing code errors. This can be attributed to the fact that regularly using programming skills helps maintain and improve their proficiency.

In Fig. 5, we can see an interesting correlation between self evaluation and average scores. It seems that the students who evaluated themselves as good in programming scored better compared to the other two groups.

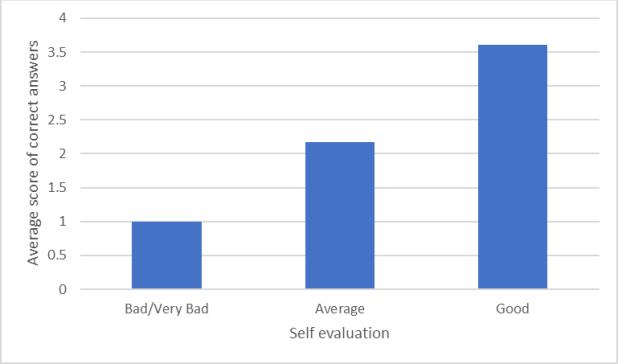

Fig. 5. Average score by self estimation.

The results suggest that experienced programmers are able to quickly identify the source of the error and implement the necessary changes, while novice programmers may struggle to understand the error message and take longer to make the correction. It is important to note that this conclusion is based on the data presented in Fig. 1 and Fig. 6 and may not necessarily apply to all cases. Factors such as the complexity of the error, the specific programming language involved, and individual aptitude for programming may also play a role in determining how quickly and accurately an error can be corrected.

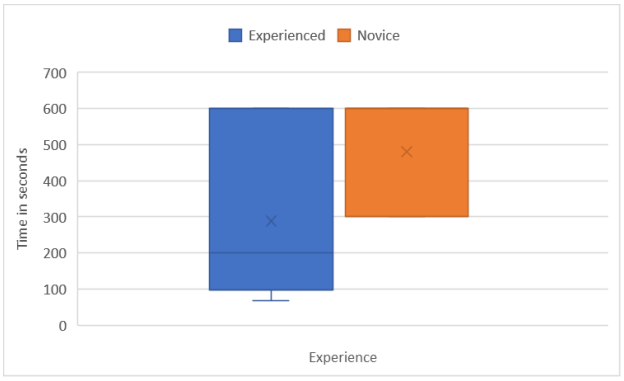

Fig. 6. Boxplot of response time of experienced and inexperienced programmers.

Significance and validity are discussed in the following sections.

IV. Discussion

A. *Significance*

The average score for older students was 3.14, while the average score for younger students was 1.25. Similarly, computer science students had an average score of 3.33, compared to 1.5 for engineering students. The study also found that students with programming experience in any language, especially in Python, performed better compared to those without experience. The self-assessment of the participants was found to be consistent with the results of the study which indicate that the self estimation is a reliable way to measure programming experience [14]. Those who regarded themselves as bad or very bad programmers on average only scored one point, while those who considered themselves to be average programmers scored 2.16 points. Those who regarded themselves as good programmers scored 3.61 points on average.

These results suggest that age and academic background, which correspond with their experience, can play a role in student's ability to correct errors in code. Additionally, having programming experience, particularly in Python, can also have a positive impact on the ability to correct errors.

In order to test the results on significance, the Shapiro-Wilk test was performed using a pre-written python script on the correctness and average response time of the two groups: Beginners and experienced programmers. The Shapiro-Wilk test is a statistical test that is used to determine if a sample of data comes from a normal distribution by comparing the sample's mean and variance to a critical value from the Shapiro-Wilk distribution table [15]. The test determined that the data did not look Gaussian for the response time for

both the beginners and experienced programmers. For the correctness, the data from the beginner group did not look Gaussian while the experienced group looked Gaussian. When the data does not look Gaussian, it means that the data is not symmetrically distributed around the mean.

Given the results from the Shapiro-Wilk test, we proceeded to apply the Mann-Whitney U-test. Mann-Whitney U-test is a non-parametric test that compares the medians of two independent samples to determine if they come from different populations [16]. According to Mann-Whitney U-test there was no statistically significant difference between beginners and experienced programmers in terms of response time; $U = 17.000$, $p = 0.135$ ($\alpha = 0.05$). However there was, indeed, a significant difference in terms of correctness; $U = 55.500$, $p = 0.025$ ($\alpha = 0.05$).

Our research question was *How does the past programming experience of the student influence the likelihood of finding the corresponding mistake in the code, and the time to find it, given the python error message?* We found that in case of correctly finding and fixing a bug the experienced group was significantly better than the novice group which was tested with Mann-Whitney U-test. Contrary to our expectations, however, there was no statistically significant difference between two groups. This would mean that our first hypothesis "*Given the same python error message, experienced programmers are more likely to find the corresponding mistake than inexperienced programmers.*" is possibly true, but our second hypothesis is likely to be false which would mean our second null hypothesis "*Given the same python error message, experienced and inexperienced programmers will find the corresponding error in the code in same amount of time.*" is possibly true.

Below is the Violin plot, which combines the box plot and also describes the data point density:

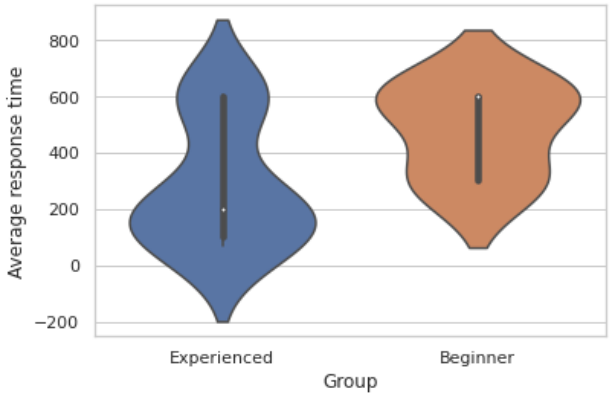

Fig. 7. Violin plot for response time.

With the violin plots we can see that both groups needed relatively same amount of time to solve the error. We can see a small concentration of experienced students between 100 seconds and 230 seconds, on the other hand, a small concentration from the novice group can be seen between 500 seconds and 630 seconds. However, as there was no significant difference between the two distribution, and due to the low number of data we were left with we cannot certainly.

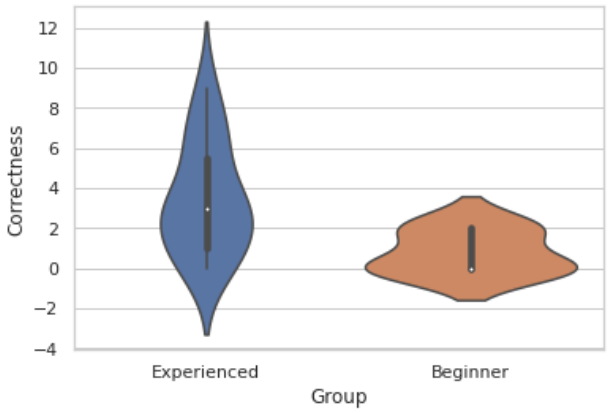

Fig. 8. Violin plot for correctness.

It is important to note that this data only shows correlation and not causation. Causation between data points shows that one collection of data or events directly influences another, for example the calories consumed per day by a group of people on average directly cause their weight to increase or decrease. Correlation, on the other hand, simply points to the fact that two data points seem to have a relationship, but this relationship is not necessarily causal.

B. Threats to Validity

Regarding an empirical research, its validity should be discussed. However, in case of external and internal validity it is often difficult to balance them as increase in external validity can lead to decrease in internal validity. Thus, it should be decided whether to concentrate on internal validity, control the experiment settings and focus on chosen aspects at a cost of generalizability, or to concentrate on external validity and observe general effects without exactly knowing which elements are indeed producing the observed results [17]. In terms of validity, following can be said about the external and internal validity of the study:

1) Internal Validity: Internal validity tries to answer the question of how strongly our study can demonstrate causation, and if our manipulation explains our outcomes. As with many other experiments, our experiment has variables that could influence the dependent variables, time and likelihood to debug the code. These factors are known as confounding variables, and they can make determining the true cause-and-effect relationship between the independent and dependent variables difficult. Confounding variables are, for example, the participant's mood at the time, distractions in the environment, their focus level, what food they ate before the survey and so on. If the participant is feeling stressed or distracted during the survey, their ability to focus and interpret the error messages correctly could be impacted. Similarly, if a participant had a heavy meal before taking the survey, their motivation could be affected. All these variables and more cannot be fully

controlled but only partly limited. These confounding variables can make it difficult to draw accurate conclusions from our study, as it can be challenging to determine whether any observed changes in the dependent variables are due to the independent variable (past experience) or due to the influence of other factors. We tried to design our experiment in such way and limit possible variables so that our internal validity would not be low. However, it should be noted, that our internal validity could not be maximized to balance out the above mentioned trade-off between internal and external validity.

2) External validity: In case of external validity, we try to answer the question of how confidently we can generalize our findings, and whether our study is representative of real life contexts. Since our study is online, we can sample a wide variety of people from different backgrounds, ages, countries, genders and so on. One advantage of conducting our study online is that it allows us to reach a much larger and more diverse population of participants than we would be able to with a traditional, in-person study. By making the survey accessible to anyone with an internet connection, we can sample participants from a wide range of backgrounds, ages, countries, and genders, which can help to ensure that our results are representative of the general population.

This is particularly important for our study, as we are investigating the relationship between past experience in programming languages and the ability to interpret error messages, and we want to make sure that our results are generalizable to a wide variety of people. By sampling a diverse group of participants, we can help to ensure that our results are not influenced by biases or other factors that may be specific to a particular demographic.

V. Conclusion and Future Work

In conclusion, the results support the hypothesis **H1**, experienced programmers are more likely to correctly address errors in code compared to novice programmers. However, the study did not find a significant difference in the time required to correct the errors between the two groups, which would mean that the hypothesis **H2** does not holds. This shows us, past experience indeed allows the programmer to correct an error but does not accelerate the solving process. It would seem surprising if a mathematics professor were to solve a problem in same amount of time as a high school student. Therefore, it is interesting for us to observe this kind of outcome in case of programming. Nevertheless, limitations such as lower number of samples than expected might have had an effect on the results. Therefore, it needs further inspection.

Future research in this area should aim to expand the scope and reach more conclusive results. One way to achieve this would be to increase the number of participants in the study. This would allow for a more diverse sample and a better understanding of the relationships between experience, error messages, and debugging speed and performance.

Another avenue for future work would be to explore other programming languages beyond Python. Python is a widely used language, but it is only one of many programming languages in use today. By examining other languages, such as Java, a more comprehensive understanding of the relationships between experience and debugging performance can be gained. This would provide a more complete picture of the impact of error messages on the debugging process and can help in identifying the best practices that can be applied across different languages.

References

[1] J. Leinonen, A. Hellas, S. Sarsa, B. Reeves, P. Denny, J. Prather, and B. A. Becker, "Using large language models to enhance programming error messages," in *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1*, 2023, pp. 563–569.

[2] B. A. Becker, P. Denny, R. Pettit, D. Bouchard, D. J. Bouvier, B. Harrington, A. Kamil, A. Karkare, C. McDonald, P.-M. Osera *et al.*, "Compiler error messages considered unhelpful: The landscape of text-based programming error message research," *Proceedings of the working group reports on innovation and technology in computer science education*, pp. 177–210, 2019.

[3] B. A. Becker, K. Goslin, and G. Glanville, "The effects of enhanced compiler error messages on a syntax error debugging test," in *Proceedings of the 49th ACM Technical Symposium on Computer Science Education*, 2018, pp. 640–645.

[4] B. A. Becker, "An effective approach to enhancing compiler error messages," in *Proceedings of the 47th ACM Technical Symposium on Computing Science Education*, 2016, pp. 126–131.

[5] Z. Zhou, S. Wang, and Y. Qian, "Learning from errors: exploring the effectiveness of enhanced error messages in learning to program," *Frontiers in Psychology*, vol. 12, p. 768962, 2021.

[6] G. Marceau, K. Fisler, and S. Krishnamurthi, "Measuring the effectiveness of error messages designed for novice programmers," in *Proceedings of the 42nd ACM technical symposium on Computer science education*, 2011, pp. 499–504.

[7] T. Kohn, "The error behind the message: Finding the cause of error messages in python," in *Proceedings of the 50th ACM technical symposium on computer science education*, 2019, pp. 524–530.

[8] N. Peitek, S. Apel, C. Parnin, A. Brechmann, and J. Siegmund, "Program comprehension and code complexity metrics: An fmri study," in *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*. IEEE, 2021, pp. 524–536.

[9] Y. Tsugawa, J. P. Newhouse, A. M. Zaslavsky, D. M. Blumenthal, and A. B. Jena, "Physician age and outcomes in elderly patients in hospital in the us: observational study," *bmj*, vol. 357, 2017.

[10] H. N. Garb, "Clinical judgment, clinical training, and professional experience." *Psychological bulletin*, vol. 105, no. 3, p. 387, 1989.

[11] G. Marceau, K. Fisler, and S. Krishnamurthi, "Mind your language: on novices' interactions with error messages," in *Proceedings of the 10th SIGPLAN symposium on New ideas, new paradigms, and reflections on programming and software*, 2011, pp. 3–18.

[12] A. Stefik and S. Siebert, "An empirical investigation into programming language syntax," *ACM Transactions on Computing Education (TOCE)*, vol. 13, no. 4, pp. 1–40, 2013.

[13] M. C. Jadud, *An exploration of novice compilation behaviour in BlueJ*. University of Kent (United Kingdom), 2006.

[14] J. Siegmund, C. Kästner, J. Liebig, S. Apel, and S. Hanenberg, "Measuring and modeling programming experience," *Empirical Software Engineering*, vol. 19, pp. 1299–1334, 2014.

[15] M. B. Wilk and R. Gnanadesikan, "Probability plotting methods for the analysis for the analysis of data," *Biometrika*, vol. 55, no. 1, pp. 1–17, 1968.

[16] H. B. Mann and D. R. Whitney, "On a test of whether one of two random variables is stochastically larger than the other," *The annals of mathematical statistics*, pp. 50–60, 1947.

[17] J. Siegmund, N. Siegmund, and S. Apel, "Views on internal and external validity in empirical software engineering," in *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*, vol. 1. IEEE, 2015, pp. 9–19.