Bachelor Degree Project

Investigating Methods to Accelerate the Solving Process of Errors and Warnings Generated by Kotlin Compilers

Author: Ege Kayıhan
Supervisor: Tobias Ohlsson
Semester: HT 2022
Subject: Computer Science

Abstract

The efficiency of error messages that are generated from compilers is up to a debate by a lot of programmers. Some believe that by enhancing the error messages, it would be easier to handle errors faster. In this paper, the ways to enhance compiler error messages and the efficiency of the found solutions for Kotlin programming language were investigated. To do this, Kotlin programmers were interviewed about the severity of the problem and possible ways to deal with it. An example plugin project was made to describe the possible concept of solving the problem described in the compiler's error message. Afterward, the same programmers were interviewed again to discuss the efficiency and practicality of the plugin. With the last interviews being done, it was seen that this plugin idea has the potential to help the computer science society and the programmers by enhancing error and warning messages.

Keywords: Kotlin, Compiler, Error Messages, Warning Messages Enhancement, Plugin

Preface

In this research, I believe that I worked on a topic that is important to the computer science society. As a senior Bachelor's student, I remember the times that I suffered trying to figure out how to understand and solve errors myself as a freshman. I believe and hope that this research will be helpful to Android programmers who have just started to learn coding with Kotlin. I would like to thank the programmers who gave me the time for me to interview them twice. I would also like to thank my reader, my examiner and my peer reviewer fellow student for their valuable feedback. Finally, I would like to thank my supervisor Tobias Ohlsson, who oversaw this research and shared with me his expertise and professional comments on the matters that I worked on.

Contents

1 Introduction

This is a 15 HEC Bachelor thesis in Computer Science. It will mainly be focusing on Android (Kotlin) Programming, compiler error and warning clarity and message enhancement.

Mobile applications have gained great importance in our lives. Nearly everyone uses smart devices nowadays with apps that are used in our daily lives. But how difficult is it for programmers to code these applications? Depending on the targeted platform, the programmer uses different languages and compilers. There are a variety of combinations to choose from. Today`s mobile application programmers use compilers to turn their projects into machine code and spot errors that cause problems for their apps. However, it is up to a discussion about the clarity of the received error and warning messages. Descriptions of some errors or warnings that come up are not very explanatory to the programmer.

This research will mainly focus on Kotlin which is an Android Programming Language. This research will be about evaluating compiler errors and warnings specifically for Kotlin in order to make suggestions on how to deal with compiler messages as fast as possible. It will also investigate if these suggestions truly accelerate the process.

1.1 Background

Programming, nowadays, is conducted in integrated development environments (IDEs). An IDE is a piece of software for creating programs that incorporate standard developer tools into a single graphical user interface (GUI). IDEs have built-in compilers to spot situations that would cause problems in running the code known as errors or warnings. Programmers need to solve these errors and warnings to be able to run their code according to common standards and its intended purpose. Enhanced descriptions of these compiler outputs decrease the time spent on fixing these errors and warnings [1]. This leads to a financial impact in the manner of cost reduction on programming efforts [5].

JetBrains developed the language Kotlin in 2010 to enhance the Java Virtual Machine programming environment [2]. It is a multi-paradigm language that supports both object-oriented and functional programming, allowing

developers to combine them as is the case with the majority of contemporary languages today. Applications may be less likely to get null pointer exceptions because Kotlin supports non-nullable types. Additionally, it has extension functions, higher-order functions, and smart casting [3]. In this research, the android programming language, Kotlin will be focused to conduct the investigation of error enhancement.

1.2 Related work

Compiler error messages are frequently insufficient and frequently discouraging, which is a major obstacle for beginners learning to program [1, 6, 7, 8, 13]. Brett Becker has looked into improving compiler error messages, although few offer meaningful empirical evidence of their effectiveness [6]. Effective error message implementation in a compiler or interpreter faces numerous theoretical and practical technical difficulties [7, 13]. For example, according to research by Brett Becker, students find it difficult to identify and fix syntax issues when they are limited to the normally brief error messages offered by the average compiler [7]. The studies of Becker show that error messages require more enhancement since current error messages` descriptions are not enough for teaching new programmers about coding [1, 6, 7, 8].

Enhanced compiler error messages may positively affect programming [1, 6, 8]. In a study on programming students, error enhancement managed to reduce factors such as the number of overall errors that have been received, errors made per/by students and repeated error types per the compiler error message [6].

Apart from enhancing the error messages with explanations, you can also direct to solutions that have been found by previous programmers that received the same error. Thiselton made a tool that automatically queries errors in Stack Overflow [4]. In summary, two ways have been used in the previous research for their example plugin; either explaining the errors better or explaining how to solve errors more clearly. Both of these factors made the plugin helpful for participants who used it [4].

1.3 Problem formulation

Programmers get different types of errors and warnings from compilers, some of which are moderately difficult to understand. Understanding and solving these errors and warnings takes time. This loss of time is costly for the company or institution [5].

A compiler`s function is to transform the code to executable format and establish errors and warnings. However, showing this to the programmer in an understandable way and advising a way to solve them in a user-friendly way may be challenging for the compiler. There is a clear gap regarding this missing functionality of the compiler and the development environments [6, 7, 8]. There is also a gap in the Android programming area regarding the clarity of the error descriptions. There is not much research conducted on Kotlin compilers.

To research this gap, the following research questions will be the main focus:

RQ1	How difficult are the Kotlin compiler error messages to understand and formulate a solution for common errors according to Kotlin programmers?
RQ2	How can you enhance specific error messages in Android Studio?
RQ3	How effective do the Kotlin programmers find the plugin implementation for error enhancement?

Table 1.1: Thesis Research Questions

1.4 Motivation

The problem that is stated in this research is an obstacle for Kotlin programmers. The time it takes to understand errors and find a way to solve them takes valuable time. The wrong ways to solve errors also delay the deployment of the applications. These situations are harmful to companies, individuals and the country's economy as economical and reputational [5].

1.5 Results

The results will be presented mainly in Section 5. This research discussed possible ways to enhance errors with Kotlin programmers. After the interviews, it is proposed that using a plugin tool might be helpful for the error enhancement process. With this idea, the procedure of technique was followed and a plugin concept was coded to show the purpose more clearly. This concept showed enhanced versions of three errors that were designated as the hardest by the Kotlin programmers from the first interview round. Plugin's effectiveness was discussed afterward in the second interview round with the same Kotlin programmers.

1.6 Scope/Limitation

There will be interviews to find three of the most difficult errors only in Kotlin. The interviews will be done with three Kotlin programmers. At the end of the interviews, there will be data present for the solving process of the errors and some of the most difficult errors to solve while coding with Kotlin. However, considering the time given for this research and since multiple reasons can trigger a specific error, it may not be possible to find a faster solving method for all of the errors. That is the reason why the plugin concept will be done for three errors only. Thus this research will be aiming to find methods to accelerate the solving process of some of these errors. The only environment that the research will be conducted in is Android Studio.

1.7 Target group

Target group of this research will be Android programmers who started to use Kotlin as their programming language and Android IDE developers to realize the necessity of error enhancement. This research will be beneficial to programmers who started learning to code with Kotlin and to programmers who need to address errors quickly by having a stack overflow site suggested by the plugin as they mostly do a copy-paste of the error and lookup from google anyway.

1.8 Outline

This report is structured as follows. In Section 2, the methodological framework, research methods and ethical considerations will be discussed.

Section 3 gives detailed information on the theoretical background and discusses the knowledge gap. In Section 4, the objectives for an evaluation technique will be discussed, which will guide the design of a technique that will be included in Section 5. This section also presents and discusses, experiences and results from the demonstration and interviews. Sections 6 and 7 describe the validation of the technique. These will be analyzed and discussed further in Section 8, where also threats to the validity of the work will be discussed with a conclusion.

2 Method

In this research, interviews will be conducted with Kotlin programmers to gather the information to answer research questions 1 and 3.

2.1 Research Project

Before starting the project, detailed literature research will be conducted about compiler error description usefulness for understanding by the programmers, compiler error enhancement with different techniques for different coding languages, Kotlin programming language and Android Programming. During the project, interviews will be conducted with Kotlin programmers about the solving process of errors in general and some of the most difficult errors that can come up during programming. After the interviews, a plugin for Android Studio will be coded to enhance some of the error messages. Then another session of shorter interviews will be conducted again with the same programmers to discuss the plugin's usefulness.

2.2 Research methods

Literature research will be conducted to get a general idea about the research done in the past. Moreover, some research will also be conducted to make an Android Studio plugin. The literature research method is one of the most common initial methods to gain information about research topics, areas of research and areas of application [9].

Mainly, semi-structured interviews will be conducted to gain insight into information regarding past experiences, from the Kotlin programmers, are required. There are some situations where some answers need elaboration for the research [10]. The interviewees will be selected from members of a mobile development team of a company where the researcher did his internship in. The interviewees have been selected to form a team at the company less than a year ago, thus they have experience from different companies in their previous work. Surveys are also another method to get this information, but it would be risky to get a certain number of people to do the survey and it would be impossible to get extra elaboration about the answers [11].

2.3 Reliability and Validity

When others replicate one's work and get the same results, that is when reliability is attained. The actions and decisions made throughout this research's conduct are openly stated in order to reduce reliability concerns. Interviews will be recorded and they will be transcribed afterward. The questions that have been asked during the interviews will be the same and there will be no persuasion to any possible answers.

Regarding validity, all of the programmers that will be interviewed are assigned to different mobile apps that require Kotlin programming. Their general programming experience is in various mobile programming languages. However, opinions are relative to the problems they faced in their pasts. Some of them had dealt with some errors more than others, which might be why they classified that error as the hardest for them among all the given. There will be three interviews conducted. If the results that will be received from the interviews create high inconsistency, additional interviews will be conducted. Also, while doing the interviews, interview bias should be avoided. That is why the interviews will be conducted through an online meeting. While the questions have been asked, it will be paid attention to not recommend giving certain answers with as much objectivity as possible by not giving any leading questions.

2.4 Ethical Considerations

Ethical considerations will be paid attention to during the interviews. For confidentiality, the interviews will be recorded by a device that has a passcode only the researcher knows and they will not be shared with anyone. Generalization, which is predicated on the idea that samples are chosen at random from the population being studied, is where sampling bias originates [12]. For sampling/bias, it is important to eliminate interview bias when conducting the interviews. In general, it is acknowledged that the interviewer's personality traits play a significant role in determining the outcome of the interview. Interviewer biases might have a direct impact on the study's final findings' validity and reliability [13]. To avoid this, the interviewees are programmers with a substantial amount of professional experience. The interviews will be performed via an online meeting, and

while the questions will be posed, care will be taken to ensure that all responses will be given objectively. Finally, for participation and consent, all of the interviewees will be notified about the reasons for the interview and their consent will be taken to record the interview and use the data that they will give for the research.

3 Theoretical Background

A few themes appear to be prevalent among the numerous program improvement techniques employed by programmers (manually) and compilers (automatically). Two of them are commonly referred to as frequency reduction and precomputation, respectively and refer to groups of optimizations for detaching calculations that can be relocated to contexts in which they will be executed less frequently or in which efficiency is less important [14].

The crucial field of compiler error messages appears to have been overlooked in research on compiler design and building. The topic can be approached in some ways or from some perspectives, according to a review of the literature. The subject is still far from being fully resolved, and numerous intriguing and unresolved problems have been found that merit additional study. Real-world examples of error messages have been given, demonstrating how badly designed some of these messages can be and how this might impact a programmer's productivity or ability to learn [15]. The issues regarding compiler error messages not being clear enough will be creating the main problem focus of this research.

The type of additional information contained in the detailed error messages containing mostly technical details presented in the console does not appear to improve message comprehension, speed up mistake detection, or help beginners more effectively spot errors. Similar research for several compiler error message characteristics (such as the technicality of the error message, visual representation, messages with examples, etc.) could pinpoint the features that have a significant favorable impact on programmers. Compiler error messages might be customized for them using this information [16]. According to research by Titus Barik, more comprehensible, human-friendly errors across a range of programming languages and compilers will result from generalizable, theory-driven methods to the design and evaluation of error messages [17]. The approach of human-friendly error messages stated in this paragraph will be helpful in a way to solve the problem of this research.

4 Research project – Implementation

During the research, interviews have been conducted and a plugin has been developed. In this section, the plugin implementation will be mainly described, since interviews have been and will be discussed in detail in the other sections, there will be less information here.

When the first round of interview questions were made, the main focus was errors that made understanding for the programmer hard and ways to solve them with an additional method/tool. The default console of the Android Studio compiler gives the error descriptions as shown in Figure 4.1, Figure 4.2 and Figure 4.3; it can clearly be seen that it does not give a clear picture of what the error is for a novice programmer. When the second round of interview questions were made, the main focus was the effectiveness of the plugin and ways to improve its features.

```
Caused by: java.lang.ArrayIndexOutOfBoundsException
   at com.example.errorthrower.FirstFragment.onViewCreated(FirstFragment.kt:39)
   at androidx.fragment.app.Fragment.performViewCreated(Fragment.java:3019)
   at androidx.fragment.app.FragmentStateManager.createView(FragmentStateManager.java:551)
   at androidx.fragment.app.FragmentStateManager.moveToExpectedState(FragmentStateManager.java:261)
   at androidx.fragment.app.FragmentStore.moveToExpectedState(FragmentStore.java:113)
   at androidx.fragment.app.FragmentManager.moveToState(FragmentManager.java:1374)
   at androidx.fragment.app.FragmentManager.dispatchStateChange(FragmentManager.java:2841)
   at androidx.fragment.app.FragmentManager.dispatchViewCreated(FragmentManager.java:2777)
   at androidx.fragment.app.Fragment.performViewCreated(Fragment.java:3020)
   at androidx.fragment.app.FragmentStateManager.ensureInflatedView(FragmentStateManager.java:394)
   at androidx.fragment.app.FragmentStateManager.moveToExpectedState(FragmentStateManager.java:260)
   at androidx.fragment.app.FragmentLayoutInflaterFactory.onCreateView(FragmentLayoutInflaterFactory.java:142)
   at androidx.fragment.app.FragmentController.onCreateView(FragmentController.java:135)
   at androidx.fragment.app.FragmentActivity.dispatchFragmentsOnCreateView(FragmentActivity.java:295)
```

Figure 4.1: ArrayIndexOutOfBounds exception from Android Studio Compiler

Figure 4.2: IndexOutOfBounds exception from Android Studio Compiler

Figure 4.3: IllegalArgument exception from Android Studio Compiler

The plugin idea was to implement a tool that would automatically run when the developer builds and runs the source code. This tool would scan the output in the Logcat (logging component) and create a popup with a detailed description and propose a way to solve the error or errors that come up. Considering the given time, it was only possible to do a subset of errors.

In order to make the plugin concept work, the following steps were implemented. Firstly, in order to read the log, the AdbController.kt file was created with AndroidLogcatService class and ILogcatListener interface. Listeners were created to take the errors from the Logcat and save them. To do the initialization of the device change listener in this file, AndroidDebugBridge class was also used.

Secondly, in order to find the error, an analysis of the console output needed to be done. In LogFactory.kt object file, the IUIAction interface was used for getting the instance. A pumpLog method was coded to place the necessary

information in the popup and a findGivenError method was coded to find the error that came up in the Logcat from the ErrorType.kt enum class.

Finally, in order to create the popup with the error information, the showErrorMessage.kt file was created. In the showError method, the popup was created as a notification and in the actionPerformed method, the information was placed in the popup after getting it from the LogFactory object.

Three errors were chosen to implement this concept. An enum class was updated to store the errors with their names that come up in the received exception, descriptions and a basic way to solve them in a user-friendly way. Storing errors was easy but not the main challenge in the process. Making the plugin a way that it would run itself after the source code ran automatically and being able to scan Logcat was the main challenge. To do so, the classes that would solve this issue were researched and by so, AndroidLogcatService and AndroidDebugBridge classes were used. By using these classes a test environment for the plugin was also able to be created. The directory of the Android Studio's binary file was given in the "runIde" section of the "build.gradle" file to create this environment. Finally, the application looks like in Figure 4.4 below.

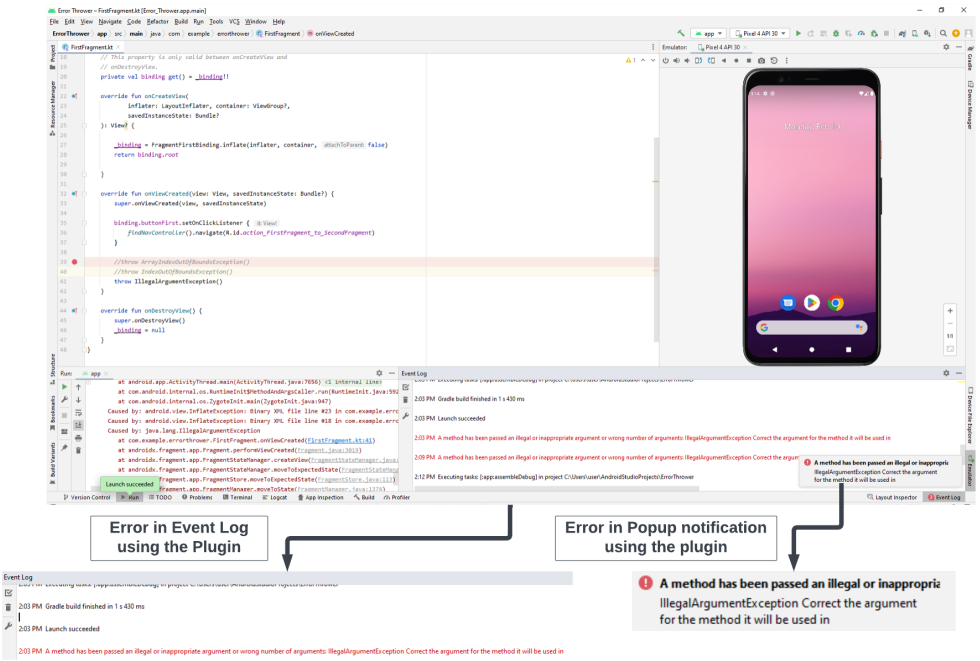


Figure 4.4: IllegalArgument exception from Android Studio Compiler

5 Results

5.1 RQ1: First Round of Interviews

The first step in the research was to get data by talking to programmers who are programming with Kotlin. Three interviews have been conducted on matters of error analysis, enhancement and difficulties and they have been transcribed and they are presented in Appendix 1 as Interview Round 1.

In the first interview, the interviewee said that the algorithms they use to deal with the errors might not be compatible with the components they use. This means they can encounter an error where it is the least expected. These situations make the solving process longer. With time, they learn to take precautions that are useful until new technology and developments make these precautions unusable and/or inefficient. The interviewee said that they would expect this situation to be solved by the IDE developers. The interviewee also said that they believe “Field/method confusion”, “Array out of bounds” and “Incorrect for-loop statement” errors are the hardest ones to solve.

In the second interview, the interviewee said that not knowing how to debug made error understanding and solving harder than it was supposed to be. With time, they decreased their need to use the internet for this process. They said special search queries with Stack Overflow integration might be a good idea to accelerate the process. They believe “Field/method confusion”, “Variable misspelled or not declared” and “Incorrect for-loop statement” are the hardest errors to solve.

In the third interview, the interviewee said that when they are dealing with an open dialogue or a fragment, which are major parts while programming in Kotlin, it is harder to spot the problems. They said that, especially in Kotlin, gradle file issues are harder to solve considering the lack of elaboration in the build feedback. They also said that some errors do not allow them to build the project either. They believe “Field/method confusion”, “Array out of bounds” and “Variable not initialized” are the hardest errors to solve.

5.2 RQ2: The Plugin and Error Enhancement

The implementation of the plugin was described in detail in Section 4. With the plugin, it has been managed to create warnings as shown in Figure 5.1, Figure 5.2, Figure 5.3, Figure 5.4, Figure 5.5 and Figure 5.6 below. The repository that has the plugin code can be found in the Plugin section in Appendix 1.

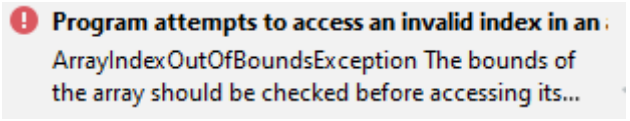


Figure 5.1: ArrayIndexOutOfBounds exception as a popup warning

Figure 5.2: ArrayIndexOutOfBounds exception in events section

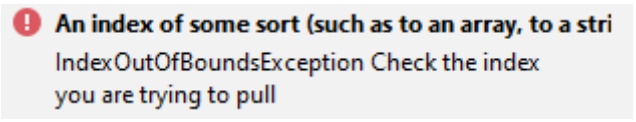


Figure 5.3: IndexOutOfBounds exception as a popup warning

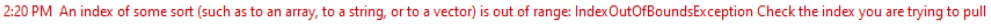


Figure 5.4: IndexOutOfBounds exception in events section

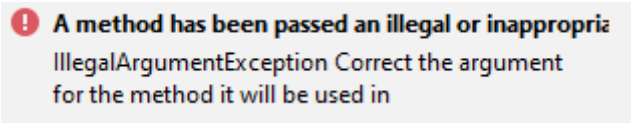


Figure 5.5: IllegalArgument exception as a popup warning

Figure 5.6: IllegalArgument exception in events section

5.3 RQ3: Second Round of Interviews

In these interviews, the plugin was discussed with the Kotlin programmers from the first interview round about its practicality and ways to improve it. These interviews have been transcribed and they are presented in Appendix 1 as Interview Round 2.

In the first interview, the interviewee found the plugin implementation useful but believed that there would be some features that could be added to make the plugin more practical for error enhancement. Some of the features that could be added for them were, forwarding to a Stack Overflow page and making a separate window inside Android Studio that shows possible solutions for the problem at hand.

In the second interview, the interviewee found the plugin implementation useful and they believed that more features could be added to make the plugin more useful for error enhancement. They believed that forwarding to the designated Stack Overflow page would be quite useful. Also checking the other questions from Stack Overflow using the same tags of the current problem would make the plugin more practical. They also thought that clicking the error name in the popup and forwarding it to the error page on Android Studio's website would also be a nice feature. Finally showing the previous users' methods to solve the error would be useful as well for the Kotlin programmers.

In the third interview, the interviewee found the plugin implementation practical and gave ideas to make the plugin better. They assumed that the aim would be for the programmer to use the console less, so they believe that the popup should also show the line where the error is generated. Other than that, they thought a button could be added to the popup to automatically search the received error from the developer's website and forward it to that page. They also thought that showing the verified comment from the Stack Overflow page that is about the received error would also be a great feature to add. Finally, showing an example of usage of the component, loop or iteration like SwiftUI, which is an IOS coding language, would also be a good idea.

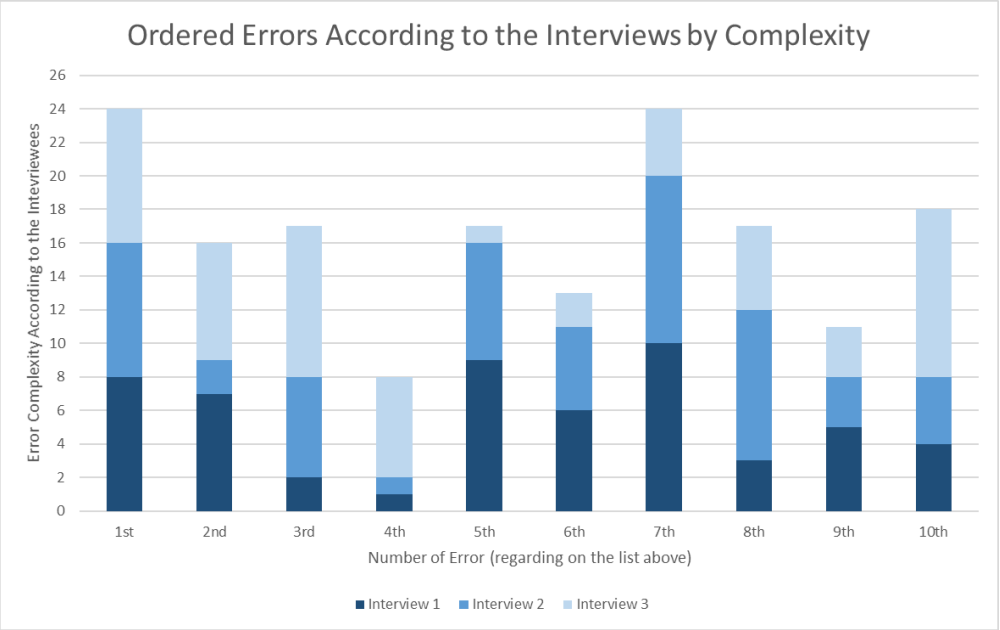
6 Analysis

This section will provide the analysis for the research.

6.1 Before the Plugin

For analysis, each interview in both of the interview rounds were transcribed. The transcription is presented in Appendix 1. According to the interviews in the first round; when they first started interviewees had problems with debugging, finding the reasons for the errors and finding solutions to errors. All of them thought that using search queries in Stack Overflow would be a solution that would speed up the process. To establish the errors to focus on for the plugin, an analysis was made to find the most complex errors by the programmers. For interview question 3, one of Brett Becker's studies was used [6]. 10 errors listed below were chosen from that study. The errors were adjusted for the Kotlin programming language for the interviewees to choose from. This analysis has been graphed in Graph 6.1 below.

1 - API function misspelled
2 - Incorrect for-loop statement
3 - Braces
4 - Other Punctuation
5 - Parenthesis
6 - Variable misspelled or not declared
7 - Array out of bounds
8 - Field/method confusion
9 - Variable not initialized
10 - Class name misspelled



Graph 6.1: Ordered Errors According to the Interviews by Complexity

The interviewees thought that "API function misspelled" and "Array out of bounds" were the hardest and most complex errors to deal with and "Other Punctuation" and "Variable misspelled or not declared" errors were the easiest among all of the errors. "Field/method confusion" and "Incorrect for-loop statement" errors were also confusing and harder to solve among all of the errors. "Braces", "Parenthesis" and "Class Name Misspelled" errors are also quite high in difficulty. For the plugin, errors that are higher in difficulty and easier to enhance should have been focused on. According to the replies from the first interview round, the errors that were chosen are "Array out of bounds", "Incorrect for-loop statement" and "Field/method confusion".

6.2 After the Plugin

The plugin is supposed to make the error understanding and solving process faster than the default features that the developers currently offer. For this to happen, a concept project has been created and this example was shown to the interviewees to get their opinions. The transcription is presented in Appendix 1. According to the interviews in the second round; the interviewees found the concept practical but it could be further developed with new features. They believed that forwarding to relevant Stack Overflow pages with a link in the popup would be making the plugin more efficient.

7 Discussion

In this research, a lot of information was gained. The programmers believe that error messages can be enhanced more to accelerate the understanding and solving processes of errors. This matter is important considering it is an obstacle in learning the programming language for programmers. Getting rid of this obstacle might also help programmers to spend less time dealing with errors since one of the interviewees stated that it takes 20% - 30% of their time during a project.

The plugin that has been made to deal with this issue might be considered helpful for programmers to understand and solve the errors while they are learning how to code with Kotlin. The interviewees finding this plugin effective shows that the plugin may have a similar effect on programming students like Becker's study [1], but it has to be tested on them to be sure.

Adding a link to the Stack Overflow site could not be managed because of the given time period for the research. However, it could have been a more efficient solution on the matter according to the interviewees and Thiselton's research [4].

8 Conclusions and Future Work

In this research, it has been found that current error descriptions that have been received from the compilers are not nearly fully understandable. It can be seen that this situation could be worked on to enhance error messages in a way it would be more helpful to the programmers. By doing this plugin concept, an example was created to address the problem. With this plugin, error messages became easier to understand by the programmers. This helps Kotlin programmers to learn errors appearing, reasoning and ways to fix them. This solution generally benefits programmers, since it decreases the time they spend on errors during a project, which also benefits the company.

In the future, with extended time, the plugin could be upgraded to a more efficient tool. In this project, it was only possible to focus on three errors due to the time period allowed for the research. More errors could be added to the plugin to check for more errors. Other than that, the descriptions of the plugin's popups could be upgraded to have a link to a Stack Overflow page that is related to the errors or to the error situation to be more precise. Using AI assistance might also be a great idea. This AI can look at the code and the error message to make suggestions as well. Also, another research could be conducted by interviewing students new to programming in order to find out the problems that they face the most during their learning process. Furthermore, doing the same research on different programming languages and platforms might also be a good idea for gaining new information about the necessity and effects of error enhancement.

References

[1] Becker, Brett. “An Exploration of the Effects of Enhanced Compiler Error Messages for Computer Programming Novices.” Nov. 2015, https://doi.org/https://www.researchgate.net/profile/Brett-Becker/publication/308890128_An_Exploration_of_the_Effects_of_Enhanced_Compiler_Error_Messages_for_Computer_Programming_Novices/links/57f50eef08ae886b897f6afd/An-Exploration-of-the-Effects-of-Enhanced-Compiler-Error-Messages-for-Computer-Programming-Novices.pdf.

[2]A. Breslav, History of Kotlin, [online] Available: https://www.coursera.org/lecturelkotlin-for-java-developerslhistory-of-kotlin-K8pZr.

[3] V. Oliveira, L. Teixeira and F. Ebert, "On the Adoption of Kotlin on Android Development: A Triangulation Study," 2020 IEEE 27th International Conference on Software Analysis, Evolution and Reengineering (SANER), 2020, pp. 206-216, doi: 10.1109/SANER48275.2020.9054859.

[4] Thiselton, Emillie, and Christoph Treude. “Enhancing Python Compiler Error Messages via Stack.” 2019 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM), 2019, https://doi.org/10.1109/esem.2019.8870155.

[5] Hoopes, David G., and Steven Postrel. “Shared Knowledge, ‘Glitches,’ and Product Development Performance.” *Strategic Management Journal*, vol. 20, no. 9, 1999, pp. 837–865., https://doi.org/10.1002/(sici)1097-0266(199909)20:9<837::aid-smj54>3.0.co;2-i.

[6] Becker, Brett A. “An Effective Approach to Enhancing Compiler Error Messages.” *Proceedings of the 47th ACM Technical Symposium on Computing Science Education*, 2016, https://doi.org/10.1145/2839509.2844584.

[7] Becker, Brett A., et al. “Compiler Error Messages Considered Unhelpful.” *Proceedings of the Working Group Reports on Innovation and Technology in*

Computer Science Education, 2019, https://doi.org/10.1145/3344429.3372508.

[8] Becker, Brett A., et al. “Unexpected Tokens.” *Proceedings of the 2019 ACM Conference on Innovation and Technology in Computer Science Education*, 2019, https://doi.org/10.1145/3304221.3325539.

[9] Snyder, Hannah. “Literature Review as a Research Methodology: An Overview and Guidelines.” Journal of Business Research, vol. 104, 2019, pp. 333–339., https://doi.org/10.1016/j.jbusres.2019.07.039.

[10] Gill, P., Stewart, K., Treasure, E. et al. Methods of data collection in qualitative research: interviews and focus groups. Br Dent J 204, 291–295 (2008). https://doi.org/10.1038/bdj.2008.192

[11] Roztocki, Narcyz, and S. D. Morgan. "The use of web-based surveys for academic research in the field of engineering." Proceedings from the 2002 American Society for Engineering Management (ASEM) National Conference. 2002.

[12] Andringa, Sible, and Aline Godfroid. “Sampling Bias and the Problem of Generalizability in Applied Linguistics.” *Annual Review of Applied Linguistics*, vol. 40, 2020, pp. 134–142., https://doi.org/10.1017/s0267190520000033.

[13] Salazar, Mary Kathryn. “Interviewer Bias.” *AAOHN Journal*, vol. 38, no. 12, 1990, pp. 567–572., https://doi.org/10.1177/216507999003801203.

[14] Jørring, Ulrik, and William L. Scherlis. “Compilers and Staging Transformations.” *Proceedings of the 13th ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages - POPL '86*, 1986, https://doi.org/10.1145/512644.512652.

[15] V. Javier Traver, "On Compiler Error Messages: What They Say and What They Mean", Advances in Human-Computer Interaction, vol. 2010, Article ID 602570, 26 pages, 2010. https://doi.org/10.1155/2010/602570

[16] Nienaltowski, Marie-Hélène, et al. “Compiler Error Messages.” *ACM SIGCSE Bulletin*, vol. 40, no. 1, 2008, pp. 168–172., https://doi.org/10.1145/1352322.1352192.

[17] Barik, Titus, et al. “How Should Compilers Explain Problems to Developers?” *Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 2018, https://doi.org/10.1145/3236024.3236040.

A Appendix 1

Interview Round 1

Interview Questions

- Are the current error messages; displayed by Android Studio as console output, clear enough?

- Do you have an idea on how to improve the error-solving process?

- Which are the hardest errors to understand and solve that are displayed by Android Studio?

  1 - API function misspelled
  2 - Incorrect for-loop statement
  3 - Braces
  4 - Other Punctuation
  5 - Parenthesis
  6 - Variable misspelled or not declared
  7 - Array out of bounds
  8 - Field/method confusion
  9 - Variable not initialized
  10 - Class name misspelled

Interview 1

1) Errors are one of the things that us programmers deal with the most. The algorithms we use to deal with these might not be compatible with the components we use, which means we can come by an error where we expect the least. Since we have deadlines that we must strictly follow, if the dealing process with errors is way too long, it creates a decrease in our motivation. Many colleagues I talked to and work with, think the same on this matter. However, developing technology and community help online (Stack Overflow, Git Hub Discussion part) brings good solutions with it. You also gain experience with going through these situations which helps you in the future such as expecting potential errors in certain places.

One of the main reasons that the solving process of these errors is so long is when you search for these errors on Google, it gives you a programmer´s comment after the code that he/she wrote and we do not see the factors that trigger the error in the background. Most of the time shared solutions do not help our situation because of the different algorithms and infrastructures we work with. I spent a lot of time on the errors I received (nearly 4-5 days) when I was a junior developer.

In a development cycle, you spent %20 - %30 of your time dealing with errors. You start to take precautions with some of the errors with experience, but newly developing technology changes the situation all the time. For example, a couple of years ago "Null Pointer Exception" was a thing in our lives and there are a lot of articles and books written about it. With new technology, this error is nearly out of our lives.

2) If you ask me, all of the developers thought about this at some point in their lives. However, if you ask me I expect IDE developers and/or component developers to handle this issue. Some IDEs started to make AIs to handle these situations which predict errors while writing the code and recommend solutions.

3) 8-7-2-1-9-6-10-3-5-4

"Field/method confusion" might be triggered by any reason. "Array out of bounds", "Incorrect for-loop statement" and "API function misspelled" takes quite a bit of time in dynamically created arrays (while inserting data in the array and receiving data from APIs). Most of the IDEs show "Other Punctuation" as a lint error which makes it the less difficult one.

Interview 2

1) When I was learning Kotlin newly and especially when I did not know how to debug, I had times when I was a complete stranger to the error. With experience, I have come to a point where I do not need to search the errors on the internet anymore. Before I started to understand the errors I had times when I did not know how to search for and solve an error as well.

2) Special search queries could be made to directly search a specific error or forward it to the correct Stack Overflow page. An integration with Stack Overflow makes sense since we programmers mostly use it while solving our errors.

3) 8-2-6-1-7-5-10-9-3-4

Punctuation and parenthesis errors in general have an easy and understandable solution suggested by the IDE and you can also assume if they will appear in certain places with experience before running.

Interview 3

1) As an experienced programmer while you are dealing with an open dialogue or fragment, you understand the error but you do not where the problem is.

2) In Kotlin, usually gradle problems are harder to solve because the console feedback is not explaining enough and it does not tell you what to do. I think a lot of Kotlin programmers suffer from the same situation. After some errors, the compiler does not let you build the code as well.

3) 8-7-9-6-1-2-4-5-3-10

The errors where you can jump to from the console after the build is easier to solve. However, those same errors can also be viewed before the build. For example, the "Array out of bounds" error requires that you need to insert inside the array, remove elements or etc. to understand the actual problem.

Interview Round 2

Interview Questions:

Did you find the plugin useful for error enhancement?

What more features can be added to make the plugin more useful?

Interview 1

1) I found this plugin concept quite useful, however, I believe there are some things that could be added to make the plugin more practical such as in the solving side.

2) Forwarding to a Stack Overflow page would be more useful. Other than that, making a separate window inside Android Studio that shows possible solutions for the problem at hand would be useful.

Interview 2

1) I found the plugin concept nice. I usually use one of the bars at the top of the IDE and event log to check the exceptions. That is why I liked this concept.

2) I believe forwarding to the designated Stack Overflow page would be quite useful. I also think that checking the other questions from Stack Overflow using the same tags of the current problem would be useful. I also thought that clicking the exception name in the popup and forwarding it to the exceptions page on Android Studio's website would also be a nice feature. You can also show the previous users' methods to solve the exception.

Interview 3

1) I found the plugin concept nice and useful, but I believe it lacks some features that can make the plugin more useful.

2) I am assuming the aim would be for the programmer to use the console less, so I believe the popup should also show the line where the error is generated. Other than that, I believe a button could be added to the popup to automatically search the received exception from the developer's website and forward it to that page. This would also be faster than searching the exception in Google. I also think that showing the verified comment from the Stack Overflow page that is about the received exception would also be a great feature. You can also show an example of usage of the component, loop or iteration like SwiftUI does.

Plugin

GitHub Repository for the Plugin:
https://github.com/egekayihan/ErrorEnhancer