Improving Text-Based Java Programming Error Messages For South African Students

A Literature Review

Danny Guttmann
gttdan002@myuct.ac.za
University of Cape Town
Cape Town, South Africa

ABSTRACT

This paper reviews literature relating to the improvement of text-based compiler error messages with the goal of improving cryptic error messages that are written in English for South African students who are not English speaking. The paper answers three research questions relating to error commonality, error message enhancement techniques, and the efficacy of enhanced error messages. Approaches to answer these questions are discussed including limitations present in the reviewed literature. It was found that: the most common errors are syntactical and semantical; natural language and visual debuggers are commonly used for message enhancement; and enhanced error messages typically yield better coding and debugging performance. The paper concludes with suggestions for further research in this area, including the need for more research on non-English speaking programmers.

KEYWORDS

Compilers, Error Messages, Novice Programmers, Enhanced Error Messages

1 INTRODUCTION

Computers understand *machine code*, a binary language which comprises ones and zeros. As machine code is a difficult language for humans to understand, many programmers write in high-level languages, which are closer to natural languages (like English). Due to this practice, there is a need for *compilers*, a special type of program that reads and translates programs written in high-level languages into equivalents written in low-level languages (like *assembly* languages), which can be assembled into machine code so that computers can run the programs.

Before translating the program, the compiler checks for any syntactic mistakes in the code. If it finds any, it alerts the programmer by displaying an error message [5].

These error messages are important to programmers, because without them, they would spend a great deal of time ascertaining the cause of their programs' inability to compile. These error messages therefore expedite faster and more efficient programming.

Error messages also play an important role in educating novice programmers by alerting them to mistakes that they make [7]. Unfortunately, the error messages generated by the compiler are not always helpful to novices, because many error messages lack detail and use complicated terminology. This may be useful for expert programmers, who are more familiar with how the different types of errors are caused, but not for novices who would not have such knowledge [11]. Another difficulty experienced by novice programmers whose first language is not English is that error messages are usually written in English. This is pertinent, particularly for South African students of programming, as English is not the first language of most South Africans [12].

To address this problem, this study will consider how to improve the quality and nature of text-based compiler error messages so that they are more helpful for South African novice programmers). This study will focus on the improvement of error messages generated by Java compilers.

To begin, the study will review available literature on improving compiler error messages to establish what is currently understood about this area. The literature review will appear in the final paper, along with details including the methodology, findings, and conclusions of the project.

This paper comprises the literature review. It outlines the datasets that have been used for testing, the different programming languages considered, the models and testing methodologies utilised, the evaluation of these models and methodologies, and the nature of the test subjects. It discusses these items and sets out conclusions.

2 MODEL AND METHODS UTILISED

It is evident from the reviewed literature that there are three distinct approaches to improve the quality and efficacy of text-based error messages. Together these approaches form a process.

The first approach is to identify the most common errors and therefore the most common error messages encountered by novice programmers [16]. This usually involves performing some sort of statistical analysis on a dataset and modelling the data as a visualisation or a taxonomy (a classification technique) [9].

The second approach is to determine how these commonly encountered error messages can be enhanced so that they are more useful to novice programmers [4, 19]. To do this, researchers may develop a novel development environment that produces enhanced error messages, sometimes in a more visual or interactive format [4].

The third step is to test whether the enhanced messages are useful to programmers and lead to better performance [18]. This is often done by giving subjects erroneous code that they did not write along with enhanced error messages and measuring the time they take to debug the code [8].

Each study reviewed takes on one or more of these approaches. This raises the three research questions (RQ) that this literature review will seek to answer.

RQ1: What are the most common types of error messages encountered by novices?

RQ2: What has been done to enhance the quality and usefulness of these error messages?

RQ3: Have novices shown an improvement in their coding/debugging ability when presented with these enhanced error messages?

3 DATASETS USED BY RESEARCHERS

A review of the datasets that have been used in previous research conducted in improving text-based compiler error messages reveals where to find datasets that are useful for researchers depending on their focus of study.

3.1 Datasets for Identifying Common Errors

The Blackbox Project is a large database that contains data collected from users of BlueJ, a Java development environment that is designed for programming students. The database contains erroneous code and the error messages that were given when the users tried to compile the code [16]. The Blackbox Project is a good source of data because it contains a large set of data from many thousands of users who have agreed to contribute their erroneous code for research purposes [9]. If a study was focusing on identifying common errors encountered by novices, the Blackbox Project would be a useful data source to consider because it provides a large sample of the different error messages encountered by novices. [13, 16].

3.2 Datasets for Error Message Enhancement Purposes

Prutor is a system that stores each version of code that has been given to the compiler, as well as snapshots of the code at regular intervals. Prutor provides a way to track novices' progress [2]. Prutor would be a good choice from which to obtain data if one was trying to determine ways of enhancing text-based compiler error messages as Prutor would be able to give a great deal of detail as to the natural of error messaging with which novices struggle as well as the coding habits of novices.

Another data source to consider would be GitHub, a web-based repository that contains large amounts of source code for huge numbers of open-source (freely available) projects of high quality. This is particularly useful if one is trying to generate improved code as opposed to simply displaying an error messages, as one could use the code on GitHub to train Machine Learning models or just to create some sort of template for working code [1].

If one does not need to identify the coding habits of novices or generate improved code, Stack Overflow should be considered. Stack Overflow is a web-based question and answer forum, which allows users to ask and answer any programming related questions. Due to the vast number of questions and answers on Stack Overflow [18], one could develop a system that queries the Stack Overflow database in order to suggest fixes for error messages and use this data to generate more detailed and more specific error messages.

3.3 Datasets for Testing Enhanced Error Messages

If a study is seeking to determine whether a set of enhanced error messages actually improves coding performance, it is preferable that the study conducts tests on programmers in order to gather data [7, 8, 17].

4 SUPPORTED LANGUAGES

The reviewed literature contained studies involving the following programming languages: Java, C, C++, C#, Python, Ada, Scheme, Eiffel, Swift, Obj. C, and OCaml. [3, 6, 17]

The most common language in the reviewed literature was Java. 65% of the studies (13 out of 20) involved the language. This is likely due to the vast amount of data available on Java, including those available from the Blackbox Project [9, 13, 16], as well as Java's pervasiveness in industry and in education [7].

5 SYSTEM/MODEL EVALUATION

The literature reviewed took different approaches to evaluate the models described above. The ways in which each model was evaluated by the various studies is detailed below:

5.1 Evaluation of the Most Common Error Messages Encountered

In order to determine which error messages are encountered by novices the most, studies tend to obtain error message data from large datasets, such as the Blackbox Project, and perform a statistical analysis on the data [16]. The statistical analysis typically includes a category development process in which different categories of errors are decided upon and a category hierarchy is formed. A process of categorisation is then done, assigning each error message to a category. From here a data analysis can be completed by calculating error frequencies, levels of error message coverage and error difficulties and severity [15]. This process may differ among studies, where some may begin with a data filtering process and end by removing outliers [13] (Figure 1 below provides a visualisation of this process). It may also use machine learning techniques to analyse data, such as Hierarchical Clustering, Support Vector Machines, Probabilistic Topic Modelling and/or Multi-Label Classification (whereby the process of hierarchy generation and categorisation is automated) [1].

The results varied slightly from paper to paper, but the general consensus was that syntactic (like missing semi-colons) and semantic (like undeclared variables) tended to be the most common types of errors among novices [13, 16].

5.2 Evaluation of Error Message Enhancement Strategies

In order to enhance error messages, studies either produce a set of guidelines based on their own or others' research [5] or they apply known principles for good error messaging and produce their own enhanced error messages [4, 14].

Guidelines discussed in the papers include Increased Readability, Reduced Cognitive Load, Provide Context, Use a Positive Tone, Provide Examples, Offer Hints or Solutions [5, 9, 19].

The reviewed literature discusses various ways in which these guidelines have been applied. Similarities in the approaches include the creation of visual debuggers which explain the cause of errors

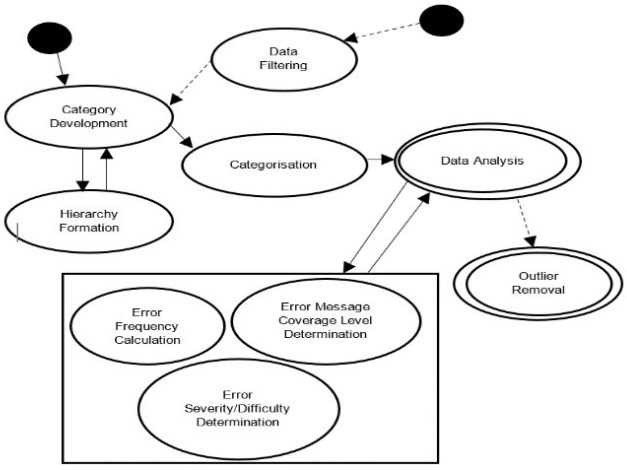

Figure 1: Visualisation of the Statistical Analysis Process for Identifying Most Commonly Encountered Errors

in a visual format [4, 14] or by rewriting the error messages in more natural language [7, 10, 11].

5.3 Evaluation of the Efficacy of Enhanced Error Messages

In order to determine whether enhanced error messages improve the coding and debugging performance of novice programmers, testing needs to be done on novices to measure the efficacy of the error messages.

The testing discussed in the reviewed literature often includes giving test subjects erroneous code to debug. The intervention portion of the subjects is given enhanced error messages and the control portion is given error messages that have not been enhanced. The researchers then record the time taken by each subject to debug the code, the number of compiler submissions and the number of error messages encountered [7, 11].

6 DISCUSSION

The sections above described the various models and methodologies utilised in the reviewed literature to improve text-based compiler error messages and how these models and methods have been evaluated. It is necessary to discuss what has been learnt from this research and, in particular, what has not worked in order to guide further research in this area.

Some studies have tried to understand and categorise errors made by novices by looking at how often a compiler gives certain error messages [13, 13]. This approach has limitations because different errors can produce the same message, and different compilers can produce different messages. Some studies have indicated that a better way to categorise errors is by manually looking at them and combining both how often they occur and how difficult they are to fix. This gives a more accurate understanding of the error messages with which novices struggle the most [16].

Another problem is that to do with taxonomies. Many studies looking at the most common error messages encountered by novices form taxonomies to categorise error messages. Although these taxonomies can be helpful, there is a risk of creating too many categories, which can make the system less efficient. Some errors may not be easily categorised and require subjective judgment to resolve. Taxonomies may need to be adjusted as new languages and features are introduced, and their effectiveness may be limited by the accuracy of the error diagnostic tools used [4].

Some studies have made the point that while previous studies have agreed that some errors are more commonly encountered than others, different studies do not always agree on which specific errors are the most common [15, 16].

Another significant point is that studies on error frequency often focus on one particular programming language and therefore the results of one study may not be relevant to other programming languages [15].

With regards to enhanced error messages, it has been found that providing too much information does not always yield positive results as is what might be expected [4]. This is due to the fact that the more detail, a message contains, the longer it will take the programmer to read and therefore will increase coding/debugging time [4].

Many studies have admitted that their results may have limited implications, because they only tested a limited number of error messages, or their study contained a small sample size [1, 6, 9]. This is particularly true for studies that used the Blackbox Project for their research as the Blackbox Project only contains data from users of BlueJ [9, 13, 16].

7 CONCLUSIONS

This literature review identified three distinct approaches taken by past researchers to improve text-based compiler error messages for novice programmers and sought to answer three research questions based on these approaches.

RQ1: What are the most common types of error messages encountered by novices?

RQ2: What has been done to enhance the quality and usefulness of these error messages?

RQ3: Have novices shown an improvement in their coding/debugging ability when presented with these enhanced error messages?

Although it varied slightly from study to study, it was identified that the most common types of error messages encountered by novices are syntactical (like missing semicolons) or semantical (like using a variable that has not yet been undeclared).

Guidelines for good error messaging have been developed and these have been applied to create visual debuggers and more detailed error messages that are often written in more concise language that is more natural than the typical error messages generated by compilers so that the messages are more understandable to novices.

Testing on novices has shown that enhanced error messaging leads to faster debugging time, fewer compiler submissions, and ultimately fewer error messages encountered during programming.

The literature reviewed made no mention of novice programmers who are not English speaking and what effect, if any, the translation of text-based compiler error messages would have on their coding

and debugging abilities. This gap opens the door for further research in this area.

REFERENCES

[1] Shubham K Agrawal. 2016. Syntax errors identification from compiler error messages using ML techniques.

[2] Umair Z. Ahmed, Pawan Kumar, Amey Karkare, Purushottam Kar, and Sumit Gulwani. 2018. Compilation Error Repair: For the Student Programs, from the Student Programs. In *Proceedings of the 40th International Conference on Software Engineering: Software Engineering Education and Training* (Gothenburg, Sweden) *(ICSE-SEET '18)*. Association for Computing Machinery, New York, NY, USA, 78–87. https://doi.org/10.1145/3183377.3183383

[3] Titus Barik, Denae Ford, Emerson Murphy-Hill, and Chris Parnin. 2018. How Should Compilers Explain Problems to Developers?. In *Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering* (Lake Buena Vista, FL, USA) *(ESEC/FSE 2018)*. Association for Computing Machinery, New York, NY, USA, 633–643. https://doi.org/10.1145/3236024.3236040

[4] Titus Barik, Jim Witschey, Brittany Johnson, and Emerson Murphy-Hill. 2014. Compiler Error Notifications Revisited: An Interaction-First Approach for Helping Developers More Effectively Comprehend and Resolve Error Notifications. In *Companion Proceedings of the 36th International Conference on Software Engineering* (Hyderabad, India) *(ICSE Companion 2014)*. Association for Computing Machinery, New York, NY, USA, 536–539. https://doi.org/10.1145/2591062.2591124

[5] Brett A. Becker, Paul Denny, Raymond Pettit, Durell Bouchard, Dennis J. Bouvier, Brian Harrington, Amir Kamil, Amey Karkare, Chris McDonald, Peter-Michael Osera, Janice L. Pearce, and James Prather. 2019. Compiler Error Messages Considered Unhelpful: The Landscape of Text-Based Programming Error Message Research *(ITiCSE-WGR '19)*. Association for Computing Machinery, New York, NY, USA, 177–210. https://doi.org/10.1145/3344429.3372508

[6] Brett A. Becker, Paul Denny, James Prather, Raymond Pettit, Robert Nix, and Catherine Mooney. 2021. Towards Assessing the Readability of Programming Error Messages. In *Proceedings of the 23rd Australasian Computing Education Conference* (Virtual, SA, Australia) *(ACE '21)*. Association for Computing Machinery, New York, NY, USA, 181–188. https://doi.org/10.1145/3441636.3442320

[7] Brett A. Becker, Graham Glanville, Ricardo Iwashima, Claire McDonnell, Kyle Goslin, and Catherine Mooney. 2016. Effective compiler error message enhancement for novice programming students. *Computer Science Education* 26, 2-3 (2016), 148–175. https://doi.org/10.1080/08993408.2016.1225464 arXiv:https://doi.org/10.1080/08993408.2016.1225464

[8] Brett A. Becker, Kyle Goslin, and Graham Glanville. 2018. 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* (Baltimore, Maryland, USA) *(SIGCSE '18)*. Association for Computing Machinery, New York, NY, USA, 640–645. https://doi.org/10.1145/3159450.3159461

[9] Brett A. Becker, Cormac Murray, Tianyi Tao, Changheng Song, Robert McCartney, and Kate Sanders. 2018. Fix the First, Ignore the Rest: Dealing with Multiple Compiler Error Messages. In *Proceedings of the 49th ACM Technical Symposium on Computer Science Education* (Baltimore, Maryland, USA) *(SIGCSE '18)*. Association for Computing Machinery, New York, NY, USA, 634–639. https://doi.org/10.1145/3159450.3159453

[10] Arthur Chargué raud. 2015. Improving Type Error Messages in OCaml. *Electronic Proceedings in Theoretical Computer Science* 198 (dec 2015), 80–97. https://doi.org/10.4204/eptcs.198.4

[11] Paul Denny, James Prather, and Brett A. Becker. 2020. Error Message Readability and Novice Debugging Performance. In *Proceedings of the 2020 ACM Conference on Innovation and Technology in Computer Science Education* (Trondheim, Norway) *(ITiCSE '20)*. Association for Computing Machinery, New York, NY, USA, 480–486. https://doi.org/10.1145/3341525.3387384

[12] Saifaddin Galal. 2022. Statista. https://www.statista.com/statistics/1114302/distribution-of-languages-spoken-inside-and-outside-of-households-in-south-africa/. Accessed: March 21, 2023.

[13] Ioannis Karvelas, Joe Dillane, and Brett A. Becker. 2020. Compile Much? A Closer Look at the Programming Behavior of Novices in Different Compilation and Error Message Presentation Contexts. In *United Kingdom amp; Ireland Computing Education Research Conference*. (Glasgow, United Kingdom) *(UKICER '20)*. Association for Computing Machinery, New York, NY, USA, 59–65. https://doi.org/10.1145/3416465.3416471

[14] Tobias Kohn and Bill Manaris. 2020. Tell Me What's Wrong: A Python IDE with Error Messages. In *Proceedings of the 51st ACM Technical Symposium on Computer Science Education* (Portland, OR, USA) *(SIGCSE '20)*. Association for Computing Machinery, New York, NY, USA, 1054–1060. https://doi.org/10.1145/3328778.3366920

[15] Davin McCall and Michael Kolling. 2014. Meaningful categorisation of novice programmer errors. In *2014 IEEE Frontiers in Education Conference (FIE) Proceedings*, Vol. 2015-. IEEE, 1–8.

[16] Davin McCall and Michael Kölling. 2019. A New Look at Novice Programmer Errors. *ACM Trans. Comput. Educ.* 19, 4, Article 38 (jul 2019), 30 pages. https://doi.org/10.1145/3335814

[17] Marie-Hélène Nienaltowski, Michela Pedroni, and Bertrand Meyer. 2008. Compiler Error Messages: What Can Help Novices?. In *Proceedings of the 39th SIGCSE Technical Symposium on Computer Science Education* (Portland, OR, USA) *(SIGCSE '08)*. Association for Computing Machinery, New York, NY, USA, 168–172. https://doi.org/10.1145/1352135.1352192

[18] Emillie Thiselton and Christoph Treude. 2019. Enhancing Python Compiler Error Messages via Stack Overflow. *CoRR* abs/1906.11456 (2019). arXiv:1906.11456 http://arxiv.org/abs/1906.11456

[19] V. Javier Traver. 2010. On Compiler Error Messages: What They Say and What They Mean. *Advances in human-computer interaction* 2010 (2010), 1–26.