Usability of Error Messages for Introductory Students

Paul A. Schliep
Division of Science and Mathematics
University of Minnesota, Morris
Morris, Minnesota, USA 56267
schli202@morris.umn.edu

ABSTRACT

Error messages are an important tool for programmers to help find and fix errors in their code. When an error message is unhelpful it can be difficult to understand how to find and fix the mistakes. Error messages are especially critical for introductory programmers in understanding problems with their code. Not all error messages are beneficial for helping novice programmers. This paper discusses how well error messages can help introductory students resolve mistakes in their programs and what aspects make an error message more user-friendly for introductory programmers. After that, we discuss the analyses of syntax, compiler, and exception errors and their results. We will also be discussing several methodologies and programs developed to help improve the experience a novice programmer has when attempting to understand causes for errors and their results.

Will most likely be changing

Keywords

Novice programmers, usability, error messages, usability studies, compiler errors, syntax errors

1. INTRODUCTION

One of the most important foundations of computer programming is the communication between the system and the user, specifically in the error messages produced by the system. These error messages are especially important for introductory-level computer science students to help them resolve issues in their program because the error messages are the primary source for understanding what is wrong and according to Marceau, "[they] lack the experience to decipher complicated or poorly-constructed feedback" [5]. The first rule of good message design is to be sure that the error does not add confusion [3]. When a novice programmer receives an error message that they can not understand, it becomes difficult to fix their program. These difficulties often leads to frustration because the error message was either too complicated to understand or lead them down the wrong path [6], which can sometimes introduce new errors [1].

Several studies have been conducted on modern programming languages' error messages to study the effectiveness in helping novice programmers debug their program and help learn the concepts and programming languages. The results have shown that students struggle with compiler and syntax error messages [1] [8] (which we will discuss in detail in Section 2) and the general vocabulary of the error messages along with IDE-specific features such as source highlighting can be bothersome for introductory computer science students [6].

Several tools and heuristics are being developed to help address issues in error message usability and its development. The goals of these methodologies are to help introductory programmers learn the language and concepts easier. The goal of this paper is to discuss the analyses of error message design and its usability for introductory students in a classroom setting, and how these developed methodologies help improve the user experience with error messages.

This paper is divided into five sections. In Section 2 we discuss usability studies, define compiler, syntax, and exception error messages, and discuss imperative and functional programming. In Section 3 we will focus on analyses of the usability of exception messages and compiler messages for introductory students and how those analyses were performed. In Section 4 we discuss the results of those analyses. In Section 5 we discuss three methodologies developed to help improve the error message usability. The first we will discuss is Traver's heuristics for compiler message design [8]. Then, we will examine Marceau, Fisler, and Krishnamurthi's recommendations for error message design [6]. Lastly, we will explore Denny, Luxton-Reilly, and Carpenter's syntax error enhancement tool [1].

Will most likely be changing

Add image of error message example

2. BACKGROUND

In order to discuss the analyses of error messages, we need to understand several concepts on error types and usability. These concepts include Compiler errors, Syntax errors, usability studies, and Human Computer Interaction. We will also be discussing the programming languages and tools whose error messages were analyzed. This includes Racket, C++, and Java programming languages and an overview of Integrated Development Environment (IDE) and DrRacket.

UMM CSci Senior Seminar Conference, December 2013 Morris, MN.

2.1 Human-computer interaction and usability studies

The study of Human-Computer Interaction, or HCI, researches how computer technology is used, specifically on the interfaces between the user and the programs of the computer. As Traver notes, "HCI is a discipline that aims to provide user interfaces that make working with a computer a more productive, effective, and enjoyable task." [8] Much of the research analyzed in this paper is from an HCI point of view, rather than a more technical approach. So, some of the areas we will be looking at in error message desig n will be the language used in the message, accuracy and precision of the messages, and other interface elements.

In order to analyze these messages in an HCI perspective and attain qualitative and quantitative information about error message usability, a usability test or case study may be performed. A usability test is a technique often used in HCI studies to evaluate a program or product by testing it on users. A case study is a research method that closely studies a group of participants, in terms of this paper, introductory students in a class room, and collects data about participants by observations and interviews. Many of the studies performed on error messages analyzed in this paper are using a case study and usability test design.

needs work

2.2 Overview of programming languages and tools analyzed

As mentioned earlier, we will primarily be analyzing how introductory students learning programming interact with error messages in terms of usability. However, in order to discuss the research and studies done on error messages, we need to define the languages and programs used.

In subsection 3.1, we are discussing a study performed by Marceau, Fisler, and Krishnamurthi that analyzes the error messages in the Racket programming language and DrRacket integrated development environment. Racket is a member of the Lisp family of programming languages, specifically designed for students new to programming. An integrated development envirnoment, or IDE, is an application that has packaged several other programs typically consisting of a text editor, compiler, and other programs used to debug code. DrRacket is an IDE meant for writing programs in Racket typically geared toward introductory students. Although Racket and DrRacket are typically geared toward introductory students and attempt to offer helpful debugging tools, the authors of the study still noticed students struggling with debugging and understanding error messages, so Marceau, Fisler, and Krishnamurthi were interested in seeing how students responded to error messages in DrRacket [6]. A rubric for suggestions on fixing these error messages in Racket and DrRacket is discussed in subsection 4.1

In subsection 3.2, we are discussing a study by Traver on compiler error message design in C++ specifically. C++ is a widely used programming language not designed for introductory programming but is occasionally taught in introductory computer science classes, but is more designed toward system programming (such as for operating systems). In subsection 4.3, we will be discussing a system meant to enhance Syntax error messages in the Java programming language. Java is a language similar to C++ and also does not offer the same level of introducing programming to students as Racket does. Java is occasionally taught in introductory classes however.

needs work

2.3 Compiler and compiler errors

There are several types of errors a user will encounter while programming. In this paper, we will be discussing the general usability of error message design, but in order to do so, we will need to define compiler errors. A compiler is a computer program (or set of programs) that transforms source code written in a programming language (the source language) into another computer language. A compilation error refers to a state when a compiler fails to compile a piece of computer program source code, either due to errors in the code, or, more unusually, due to errors in the compiler itself. Being able to resolve compiler errors is crucial because it is the first step toward resolving issues with the program as the code will not run when there are compiler errors. Below is an example of a compiler error in C++, modified from [8]. In this code snippet, the keyword friend is used inside the declaration of a class giving friend to a particular function but not when defining that function.

```
friend ostream & operator <<
(ostream & os,constSavingAccount & sA){
os << "this is a saving account";
}

Error: can not intialize friend function '<<'
friend declaration not in class definition
```

There is also a subset of a compiler error, called a Syntax error, which is "a mistake in the syntax of some strange sequence of characters or tokens that is intended to be stated in a particular programming language." Syntax error messages are especially important for introductory students to develop their debugging skills. As Kummerfield and Kay note [4], "The usability of compiler errors are important because Syntax error correction is the first step in the debugging process. It is not possible to continue program development until the code compiles. This means it is a crucial part of the error correction process." We will be discussing Syntax error messages more in section 4. Below is an example of a Syntax error in Java (modified from [1]). In this scenario, the error message is trying to tell the user that the if statement needs to be surrounded by opening and closing parentheses.

```
if (score < 0) || (score > 100)
Syntax error on token "||", if expected
```

This paper will also briefly discuss the parser, which is a program usually part of a compiler that receives input as program instructions or markup and breaks them into parts that can be managed by other programming (such as other components in a compiler). The parser is also the program that checks for correct syntax in the process of building data structures from the inputs.

Since introductory students will be new to programming, they will be dealing with compiler and Syntax errors early on, so the usability of these error messages is an important part in order for the students to start learning the concepts. These error compiler error messages are what students will

```
(define (label-near? name bias word1 word2 word3)
  (cond
    (and (cond [(string=? name word1) "Name Located"]
               [(string=? bias word1) "Bias Located"])
         (cond [(string=? name word2) "Name Located"]
               [(string=? bias word2) "Bias Located"])
    "Mark")
))

 and: found a use of `and' that does not follow an
open parenthesis
```

Figure 1: Example of an ineffective error message in Racket

be using to debug their programs often while learning the programming language, and these messages need to be user-friendly for them to better understand how to fix their mistake and learn from it. However, compiler error messages are often cryptic and difficult to understand for many programmers, especially for students who are new to programming. Unfortunately, as Traver notes, "most related disciplines, including compiler technology, have not paid much attention to this important aspect that affects programmers significantly, apparently because it is felt that programmers should adapt to compilers." [8]

find sources to define syntax and compiler errors

3. ANALYSES

This section will discuss two different studies performed on the usability of error messages and what the results of those studies are. The first analysis will discuss how well the error messages in Racket and DrRacket help introductory students debug their programs. The second analysis will discuss the effectiveness of compiler error messages in the C++ programming language.

3.1 Analysis of error messages in Racket and DrRacket

In the spring of 2010, Marceau, Fisler, and Krishnamurthi ran a usability study on error messages in DrRacket. The study involved configuring DrRacket to save a copy of each program a student tried to run as well as the error message through 6 50 minute lab sessions [5]. The authors were interested in which error messages are effective and in what ways and how well DrRacket's text highlighting can help a student.

In order to measure effectiveness, the authors developed a rubric which determined whether the student made a reasonable edit in response to the error message [5]. The rubric was meant to distinguish how an error message would fail or succeed. They determined that an error message is effective if a student can read it, understand it, and use that information to figure out how to resolve the issue.

Figure 1 shows an example of an error message in Racket that is not effective for a student. The message is contradicting itself as it does follow an open parenthesis, but the parser thinks the `and` is an independent part from the `cond`.

Marceau, Fisler, and Krishnamurthi grouped messages into nine most common error categories in their results from

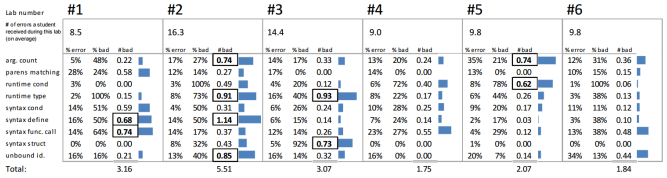

Figure 2: Results from DrRacket study

the study. Their rubric found the level of severity of the error messages. Through their data collection at the end of the study, they also found the level of occurrence of each error type from each lab and which error messages were poorly responded to, indicated as *%error* and *%bad*, respectively. *#bad* shows the level of likelihood of recurrence of the respective error message. The values of interest are the values enclosed in a box, which are the higher level *#bad* values, as seen in figure 2.

The authors found from the data, that students have difficulties with certain errors at different points in the course. For example, in Lab 5 of the course, the authors found that the numerous argument count errors trace to mistakes in properly closing expressions. The error messages that the student saw from these mistakes did not match with the actual error the student encountered with their program.

fix image positions

3.2 Analysis of compiler messages in C++

Compiler error messages are often difficult to understand for many programmers, especially introductory programmers. Not a lot of research has been conducted on compiler error message design. So, in the first semester of 2002-2003, Traver conducted a case study on students' work with compiler error messages in C++ in an introductory computer science course. The motivation of this study is to gain insight on what students are struggling with in the course and to help the professor's personal struggles with these error messages. Traver gathered data from the students' interactions with C++ throughout the semester and wrote up analyses of the error messages received in 5 separate parts.

- *The error message* received from the compiler
- *The source code* that caused the original error
- *The diagnostic* of why the error occurred
- *An alternate error message* that may help lead more directly to the true diagnosis of the issue.
- *A comment* about why the error message is not helpful

Below is an example of an error message in C++ analyzed in the study along with the source code [8] (in the interest of space, I have not included the other parts of the analysis):

```
Source Code:

class SavingAccount
friend ostream & operator<<
ostream &os, const SavingAccount &sA);
};
```

```
Error Message:

ANSI C++ forbids declaration
'ostream' with no type 'ostream'
is neither function nor method;
cannot be declared friend
parse error before '&'
```

In this case, the user forgot to include a header file (iostream.h), so the compiler does not know what iostream is. The error message however, did not suggest to include a header, a simple fix, where the original error message could easily confuse novice programmers. The author of the study noted that this type of error message should " convey a clear message that the programmer can quickly understand and that is useful for fixing the error", but the error message given to the user would not accomplish this for students who are still new to programming [8].

Traver found from the study, that there is an apparent lack of thought put into the usability of compiler error messages and that many students, especially those new to programming, will have a hard time understanding these errors. There was no quantitative data gathered on these error messages, but rather just a general observation of how students struggled working with the compiler error messages. Traver also noted that "better [error messages] are possible" and was able to provide alternative error messages for each of the ones analyzed [8]. So, better error message design is possible, such that more efforts and time are put into compiler error message research.

4. METHODOLOGIES

In this section, we will be discussing three tools and methodologies meant to attempt to improve the usability of error messages or suggest improvements for error message design based on the results of the analyses discussed in section 3. The first methodology we will discuss is a set of recommendations for improving the usability of error messages. The second approach discusses improving compiler error messages through a set of principles meant to increase usability of the messages. For the third approach, we will discuss an attempt on enhancing syntax error messages in Java and the how well these syntax error messages improve over the original.

4.1 Recommendations for error messages

This will use my source, Mind Your Language: On Novices' Interactions

TODO

4.2 Principles of compiler error design

This will use my source, On Compiler Error Messages: What They Say and What They Mean

TODO

4.3 Syntax error message enhancement and results

This will use my source, Enhancing Syntax Error Messages Appears Ineffectual

TODO

5. CONCLUSION

Discussion of the direction usability of error messages will be taken and how the methodologies will be applied for future work.

TODO

6. ACKNOWLEDGMENTS

TODO

7. REFERENCES

[1] Denny, P., Luxton-Reilly, A., and Carpenter, D. Enhancing syntax error messages appears ineffectual. In *Proceedings of the 2014 Conference on Innovation & Technology in Computer Science Education* (New York, NY, USA, 2014), ITiCSE '14, ACM, pp. 273–278.

[2] Hartmann, B., MacDougall, D., Brandt, J., and Klemmer, S. R. What would other programmers do: Suggesting solutions to error messages. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (New York, NY, USA, 2010), CHI '10, ACM, pp. 1019–1028.

[3] Isa, B. S., Boyle, J. M., Neal, A. S., and Simons, R. M. A methodology for objectively evaluating error messages. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (New York, NY, USA, 1983), CHI '83, ACM, pp. 68–71.

[4] Kummerfeld, S. K., and Kay, J. The neglected battle fields of syntax errors. In *Proceedings of the Fifth Australasian Conference on Computing Education - Volume 20* (Darlinghurst, Australia, Australia, 2003), ACE '03, Australian Computer Society, Inc., pp. 105–111.

[5] Marceau, G., Fisler, K., and Krishnamurthi, S. Measuring the effectiveness of error messages designed for novice programmers. In *Proceedings of the 42Nd ACM Technical Symposium on Computer Science Education* (New York, NY, USA, 2011), SIGCSE '11, ACM, pp. 499–504.

[6] Marceau, G., Fisler, K., and Krishnamurthi, S. 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* (New York, NY, USA, 2011), Onward! 2011, ACM, pp. 3–18.

[7] Murphy, C., Kim, E., Kaiser, G., and Cannon, A. Backstop: A tool for debugging runtime errors. In *Proceedings of the 39th SIGCSE Technical Symposium on Computer Science Education* (New York, NY, USA, 2008), SIGCSE '08, ACM, pp. 173–177.

[8] Traver, V. J. On compiler error messages: What they say and what they mean. In *Advances in Human-Computer Interaction* (2010).

Citing sources for references

[1] [2] [3] [4] [5] [6] [7] [8]