

# Usability of error messages for introductory students

Paul Andrew Schliep

Division of Science and Mathematics  
University of Minnesota, Morris  
Morris, Minnesota, USA

25 April 2015  
University of Minnesota, Morris

# Introduction to error messages

- Mistakes happen a lot in programming
  - especially for new programmers
- Error messages returned from programming mistakes
- Tells the user information about the issue
- Here's an example of an error message:

```
int three = 3;  
System.out.print(to);  
-> error:  
'three' cannot be resolved to a variable
```

# Importance of error messages

- Analyze error messages from usability perspective
- Error messages are important tool for beginner programmers
  - one of the primary interactions between the system and the user
- A good error message should:
  - be easy to understand
  - help a student locate the issue
  - not add confusion
- Unhelpful error messages create frustration

# Example of poor error message

developing...

# Outline

- 1 Background
  - Compiler and runtime errors
  - Dynamic and statically typed
- 2 Analyses of error messages
  - Analysis of DrRacket IDE
  - Analysis of compiler errors
- 3 Methodologies for improving error messages
  - Recommendations for improving IDE error messages
  - Analysis of syntax error enhancement
- 4 Conclusions

# Outline

- 1 Background
  - Compiler and runtime errors
  - Dynamic and statically typed
- 2 Analyses of error messages
- 3 Methodologies for improving error messages
- 4 Conclusions

# Compiler errors

- When a compiler fails to compile a program, a user will receive a compiler error message
- For newer programmers, these typically occur from syntax errors
- Example (in Java):

```
int seven = (2 + 5;  
error: ')' expected
```

# Runtime errors

- A runtime error occurs after a program has compiled
- Usually indication of logical errors in the code
- Cannot be predicted, dependent on the values
- Example:

```
String string = "hello";  
System.out.print(string.substring(3,6));
```

```
java.lang.StringIndexOutOfBoundsException:  
String index out of range: 6
```



# Statically typed languages

- Variables assigned types
- Type checking done at compile time
- Would give compile time error

```
String personName = "Frank";  
personName = 7;
```

# Dynamically typed languages

- Variables are not assigned to types
- Type checking done at runtime
- Would give runtime error

```
personName = "Frank"  
personName = 7
```

# Outline

- 1 Background
- 2 Analyses of error messages
  - Analysis of DrRacket IDE
  - Analysis of compiler errors
- 3 Methodologies for improving error messages
- 4 Conclusions

# Overview of study

- Marceau et al. noticed students struggling with error messages in course
- Conducted study on DrRacket error messages in Spring of 2009
- Hoping to use the data to improve students' interactions with DrRacket error messages

# Integrated development environments

- An integrated development environment (IDE) is a program for writing and running code
- Some IDEs come packaged with debugging tools and custom error messages

# Racket programming language

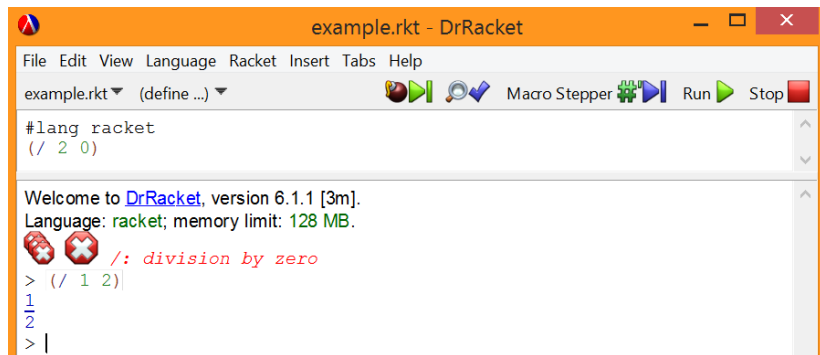
- Programming language useful for teaching in introductory courses
- Member of Lisp languages
- Functional language: computation as a composition of functions and retains immutable data and avoids changing state
- dynamically typed
- Syntax example:

```
(+ 1 2)  
-> 3
```

# DrRacket

- An IDE for developing programs in Racket
- Geared toward introductory programmers
- DrRacket offers (mostly) user-friendly error messages and libraries to program in various levels

# DrRacket interface





# Study of DrRacket error messages

- Marceau et al. interested in finding which errors students struggled with
- Configured DrRacket to save a copy of each program a student tried to execute and the error messages received
- Programs taken from a once-per-week lab session

# Table of results

Lab Number	#1			#2			#3			#4			#5			#6		
	%error	%bad	#bad	%error	%bad	#bad	%error	%bad	#bad	%error	%bad	#bad	%error	%bad	#bad	%error	%bad	#bad
arg. count	5%	48%	0.22	17%	27%	<b>0.74</b>	14%	17%	0.33	13%	20%	0.24	35%	21%	<b>0.74</b>	12%	31%	0.36
parens matching	28%	24%	0.58	12%	14%	0.27	17%	0%	0.00	14%	0%	0.00	13%	0%	0.00	10%	15%	0.15
runtime cond	3%	0%	0.00	3%	100%	0.49	4%	20%	0.12	6%	72%	0.40	8%	78%	<b>0.62</b>	1%	100%	0.06
runtime type	2%	100%	0.15	8%	73%	<b>0.91</b>	16%	40%	<b>0.93</b>	8%	22%	0.17	6%	44%	0.26	3%	38%	0.13
syntax cond	14%	51%	0.59	4%	50%	0.31	6%	26%	0.24	10%	28%	0.25	9%	20%	0.17	11%	11%	0.12
syntax define	16%	50%	<b>0.68</b>	14%	50%	<b>1.14</b>	6%	15%	0.14	7%	24%	0.14	2%	17%	0.03	3%	38%	0.10
syntax func call	14%	64%	<b>0.74</b>	14%	17%	0.37	12%	14%	0.26	23%	27%	0.55	4%	29%	0.12	13%	38%	0.48
syntax struct	0%	0%	0.00	8%	32%	0.43	5%	92%	<b>0.73</b>	0%	0%	0.00	1%	0%	0.00	0%	0%	0.00
unbound id.	16%	16%	0.21	13%	40%	<b>0.85</b>	16%	14%	0.32	16%	0%	0.00	20%	7%	0.14	34%	13%	0.44

%error: Percentage of error messages during lab of the given category of errors  
**KEY:** %bad: Percentage of error messages that were poorly responded to  
 #bad: Estimate of the number of errors in the category that each student responded poorly to

# Results

- Students struggle with certain errors relative to skill level
- Some errors were not indicator of underlying issue
  - student struggled with these errors
  - suggests issues in error message effectiveness

# Student code example

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

# Overview of study

- Compiler error messages often cryptic and difficult for many programmers
- Traver and his students found compiler errors messages difficult to understand
- Traver conducted study in Fall 2002 at Jaume I University to verify which errors intro students struggle with
  - course used C++ programming language

# Intro to C++

- Not designed to be taught in intro course
- Imperative language: uses memory manipulation and state-changing statements to build computation
- statically-typed
- Object-oriented programming (OOP): method of programming around class hierarchy and creating objects
- Syntax example:

```
int a = 2;  
a = a + 2;  
cout << a;  
-> 2
```

# Method of study

- GNU g++ compiler was used
- Code gathered from students in lab sessions throughout semester
- Analyzed each message and wrote out the following for each message:
  - why the error occurred
  - possible alternate error message
  - why the error is unhelpful

# Example of code analyzed

## Offending code:

```
SavingAccount::SavingAccount() {  
    float SavingAccount::getInterestRate() {  
        return rate;  
    }  
}
```

## Error message:

In method 'SavingAccount::SavingAccount()':  
declaration of 'float SavingAccount::getInterestRate()' :  
outside of class is not definition



# Example continued

## Alternative error message:

A function declaration inside a function body is not possible. Did you forget '}' to close the body of the previous function definition?

# Results

- Study makes a good case for compiler error usability
- Hopes that approaches be considered to improve messages
- Helped him understand which errors students students struggled with

# Outline

- 1 Background
- 2 Analyses of error messages
- 3 Methodologies for improving error messages**
  - Recommendations for improving IDE error messages
  - Analysis of syntax error enhancement
- 4 Conclusions

# Recommendations

- Marceau et al. based their research on these
- Wanted to maintain two design principles:
  - error messages should not propose solutions
  - error messages should not prompt toward incorrect edits

# recommendations continued

- Simplify message vocabulary
  - eg, student will understand `variable` more than `identifier`
  - these should be for lower levels in DrRacket
- Be explicit with constructor or function usage in highlighting
- Color coding references with its corresponding code

# Color coded error message

Red highlights definition, green highlights clause, blue highlights definition

```
;; Produces a true or false answer depending on if the label
appears within three words of the name
(define (label-near? label name word-one word-two word-three)
  (cond
    [(and (string=? "name" "word-one")
          (string=? "label" "word-two") "true")]
    [(and (string=? "name" "word-one")
          (string=? "label" "word-three") "true")]
    [(and (string=? "name" "word-two")
          (string=? "label" "word-one") "true")]
    [(and (string=? "name" "word-two")
          (string=? "label" "word-three") "true")]
    [else "false"]])
```

Annotations in the image:

- Red**: points to the `(define)` keyword.
- Green**: points to the `(cond)` keyword.
- Blue**: points to the first clause `[(and (string=? "name" "word-one") (string=? "label" "word-two") "true")]`.
- Green**: points to the closing bracket of the first clause `]`.
- Red**: points to the closing bracket of the `(cond)` block `])`.

Welcome to [DrScheme](#), version 4.2.2 [3m].

Language: Beginning Student, memory limit: 128 megabytes.

```
cond: expected a clause with a question and answer, but found a
clause with only one part
```

Annotations in the image:

- Red**: points to the `cond:` prefix.
- Green**: points to the `cond:` prefix.
- Blue**: points to the phrase `one part`.

# Future work

- Recommendations implemented in How to Design Programs (HtDP) libraries in DrRacket
- Further research needed to evaluate HtDP libraries

# Intro to syntax errors

- Introductory students most likely receive many syntax errors in first few weeks of learning language
- "Syntax errors are significant barrier to student success" - Denny et al.
- Denny et al. propose to improve syntax error messages in course



# Improving errors

- Course used Java, language similar to C++
- Course also used CodeWrite, online IDE
- Pulled student submissions from CodeWrite
- Match commonly erroneous code, extracted line containing error, and inserted their enhanced error

# Enhanced syntax error example

It appears that there is an error in the condition below:

```
if (score < 0) || (score > 100)
```

Remember that the condition for an `if` statement must be surrounded by opening and closing parentheses:

```
if (condition)
```

This is true even if the condition consists of more than one boolean expression combined with logical operators like `&&` or `||`.

## Example

### Incorrect Code

```
int a = 6;
double x = 9.4;

if (x > 10) && (a == 0) {
    return true;
}
```

### Correct Code

```
int a = 6;
double x = 9.4;

if ((x > 10) && (a == 0)) {
    return true;
}
```

## Explanation

The condition of an `if` statement needs to be enclosed in parentheses. Even if the condition is made up of the combination of other conditions, the entire thing still needs to be wrapped in parentheses

# Testing the enhanced syntax messages

- Students in control group (original error messages) and intervention group (enhanced error messages)
- Compared attempts of student submissions to see if it would reduce the number of non-compiling submissions overall
- Used t-tests to find results (still need to define t-test)

# Results of syntax enhancement

- t-tests gave high p-values ( $p > 0.05$ )
  - indication of no evidence of significant difference between the groups

# Future work

- Denny et al. believed several factors were cause for no significance
  - for example, students may not have paid attention to additional information
- Authors would like to apply additional research to improve their messages

# Outline

- 1 Background
- 2 Analyses of error messages
- 3 Methodologies for improving error messages
- 4 Conclusions**

# Results

- Some work needs to be done with DrRacket error messages to better report the actual errors
- Compiler error messages are an issue in terms of usability
- Research is being done in attempt to improve error messages
  - Marceau et al. recommendations for error message design in IDEs
  - Denny et al. syntax error message enhancement

# Future work

- HtDP libraries implemented Marceau et al. research
- Some work being done to attempt to improve compiler error messages
- Growing interest in computer science
  - more user-friendly error messages are a necessity



# Acknowledgments

I would like to thank the following people:

- My advisor, Elena Machkasova, for helping with my senior seminar and useful feedback on my paper and presentation
- Stephen Adams and Jim Hall for providing useful feedback on my paper
- Friends and family for attending
- Paul Schliep, as none of this would have been possible without him

# Thanks!

Thank you for your time and attention!

Contact:

- `schli202@morris.umn.edu`
- `github.com/Paul-Schliep`

## Questions?

# References



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.



Traver, V.J.

On compiler error messages: What they say and what they mean.

In Advances in Human-Computer Interaction, 2010.

See my senior seminar paper for additional references.