Usability of Error Messages For Introductory Students

Paul A. Schliep

Division of Science and Mathematics
University of Minnesota, Morris

25 April 2015

Introduction to error messages

- Mistakes happen a lot in programming
  - especially for new programmers
- Error messages produced from programming mistakes
- Tells the user information about the issue
- Here's an example of an error message (in Java):
  - `->` denotes output

```
int three = 3;
System.out.print(tree);

-> error:
'tree' 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:
  - convey the actual issue
  - help a student locate the issue
  - not add confusion
- Unhelpful error messages create frustration

Outline

1. Background
   - Compiler and runtime errors
   - Dynamic and statically typed
2. Analyses of error messages
   - Analysis of Racket and DrRacket
   - 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

- Compiler: converts code to lower level language so instructions can be computed
- Compiler errors occur when code cannot be compiled due to errors
- Program will not execute
- For newer programmers, these typically occur from syntax errors
- Example (in Java):

```
int seven = (2 + 5;

-> error: ')' expected
```

Runtime errors

- Runtime error occurs after a program has compiled, during execution of the program
- Dependent on values that may not be known at compile time
- Usually indication of logical errors in the code
- Example:

```
String theString = "hello";
int low = 3;
int high = 6;
System.out.print(theString.substring(low,high));

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

Statically typed languages

- Variables assigned types
- Type checking done at compile time
- Following code would give compile time error

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

Dynamically typed languages

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

```
var x = "Frank";
var y = 7;
x - y;
```

Outline

1 Background

2 Analyses of error messages
- Analysis of Racket and DrRacket
- Analysis of compiler errors

3 Methodologies for improving error messages

4 Conclusions

Racket programming language

- Racket: Programming language useful for teaching in introductory courses
- Functional language: computation as a composition of functions, retains immutable data, avoids changing state
- Dynamically typed
- Syntax example:

```
(+ 1 2)

-> 3
```

DrRacket integrated development environment

- An integrated development environment (IDE) is a program for writing and running code along with debugging tools
- DrRacket is an IDE for developing programs in Racket
- Geared toward introductory programmers
- DrRacket offers user-friendly error messages and libraries

Study of DrRacket error messages

- Marceau et al. noticed students struggling with error messages in course
- Interested in which errors students struggled with
- Conducted study on DrRacket error messages in Spring 2010
- Hoping to use data to improve student interactions with DrRacket error messages

Method

- Configured DrRacket to save a copy of each program from 6 consecutive lab sessions
- Measured errors students responded poorly to
- Group errors into nine related categories

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%	**0.74**	14%	17%	0.33	13%	20%	0.24	35%	21%	**0.74**	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%	**0.62**	1%	100%	0.06
runtime type	2%	100%	0.15	8%	73%	**0.91**	16%	40%	**0.93**	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%	**0.68**	14%	50%	**1.14**	6%	15%	0.14	7%	24%	0.14	2%	17%	0.03	3%	38%	0.10
syntax func call	14%	64%	**0.74**	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%	**0.73**	0%	0%	0.00	1%	0%	0.00	0%	0%	0.00
unbound id.	16%	16%	0.21	13%	40%	**0.85**	16%	14%	0.32	16%	0%	0.00	20%	7%	0.14	34%	13%	0.44
#bad Total:			**3.16**			**5.51**			**3.07**			**1.75**			**2.07**			**1.84**

KEY:
%error: Percentage of error messages during lab of the given category of errors
%bad: Percentage of error messages of the given error category that were poorly responded to
#bad: Estimate of the number of errors that were responded poorly to (AvgErrTotal * %error * %bad)

Student edit example

- Missing parentheses before `cond` on line 2
- Same error message produced for each edit

```
(define (string-one-of? check-for-match stringOne stringTwo stringThree)
   cond [(and (string=? check-for-match stringOne))]
        [(and (string=? check-for-match stringTwo))])
```

↪ *define: expected only one expression for the function body, but found at least one extra part*

```
(define (string-one-of? check-for-match stringOne stringTwo stringThree)
   cond [(string=? check-for-match stringOne)]
        [(and (string=? check-for-match stringTwo))]
        [(and (string=? check-for-match stringThree))])
```

```
(define (string-one-of? check-for-match stringOne stringTwo stringThree)
   cond [and ((string=? check-for-match stringOne))]
        [(and (string=? check-for-match stringTwo))]
        [(and (string=? check-for-match stringThree))])
```

Student edit example continued

```
(define (string-one-of? check-for-match stringOne stringTwo stringThree)
   cond [(string=? check-for-match stringOne)]
        [(string=? check-for-match stringTwo)]
        [(string=? check-for-match stringThree)])

(define (string-one-of? check-for-match stringOne stringTwo stringThree)
   cond [(string=? check-for-match)]
        [(string=? check-for-match stringTwo)]
        [(string=? check-for-match stringThree)])
```

Marceau et al.

- Student did not fix error
- Fixed code:

```
(define (string-one-of? check-for-match stringOne stringTwo stringThree)
  (cond [(string=? check-for-match)]
        [(string=? check-for-match stringTwo)]
        [(string=? check-for-match stringThree)]))
```

Results

- Errors that were difficult to resolve was relative to course material
- Some errors were not indicator of underlying issue
  - students struggled with these errors
  - suggests issues in error message effectiveness
- Further research on understanding actual underlying errors and improving DrRacket error messages

Analysis of compiler errors

- Students struggle with compiler error messages in course taught by V. Javier Traver at Jaume I University
- Traver interested in finding compiler error messages students struggled with
- Conducted study in Fall 2010
  - course used C++ programming language
- Study from a strictly usability perspective
  - did not consider technical constraints

Intro to C++

- C++: Programming language not designed to be taught in intro course
- Imperative language: uses memory manipulation and state-changing statements
- Statically-typed
- Syntax example:

```
int a = 2;
a = a + 2;
cout << a << endl;

-> 4
```

Method of study

- GNU g++ compiler used
- Erroneous student code gathered from semester
- Analyzed each message:
  - why the error occurred
  - possible alternative error message
  - why the error message is unhelpful

Example of code analyzed

Offending code (missing curly brace):

```
1 SavingAccount::SavingAccount(){
2     float SavingAccount::getInterestRate(){
3       return rate;
4     }
```

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?
```

- Original error is confusing
  - does not tell user missing curly brace

Results

- No quantitative data, just observations
- Hopes that approaches be considered to improve messages
- Future work: develop methods to improve messages in course

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

- Recommendations on error message design
- Marceau et al. used previous research
- 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 highlighting
- Color coding references with its corresponding code

Poor highlighting example

- Runtime error, never refers to highlighted expression

```
(define-struct culture (type div-times))
(define culture1
  (make-culture "E-Coli" (cons 20 (cons 18 (cons 16 empty)))))
(define (sum-div-time a-culture)
  (+ (first (culture-div-times a-culture)) (rest (culture-div-times
a-culture))))
(sum-div-time culture1)
```

```
Welcome to DrRacket, version 5.1 [3m].
Language: Beginning Student.
+: expects type <number> as 2nd argument, given: (cons 18 (cons 16 empty));
other arguments were: 20
>
```

Marceau et al.

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"]))
```

Blue

Green

Green

Red

Red

```
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
>
```

Blue

Green

Red

Future work

- How to Design Programs (HtDP) libraries for DrRacket
- Further research needed to evaluate HtDP libraries

Intro to syntax errors

- Syntax errors when learning a new language
- "Syntax errors can be a significant barrier to student success" - Denny et al.
- Denny et al. propose to improve syntax error messages

Improving errors

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

Syntax error

Erroneous code:

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

```
-> Syntax error on token "||", if expected
```

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 `||`.

Denny et al.

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 from course in control group (original error messages) and intervention group (enhanced error messages)
- Compared attempts of student submissions
- Used t-tests to find results
  - t-test: finds if there is statistically significant difference between two data groups

Results of syntax enhancement

- t-tests gave high p-values ($p > 0.05$)
  - indication of no evidence of statistically significant difference
  - note: not all tests shown
  - this test compares longest consecutive failing submissions

Exercise	μControl	μEnhanced	p-value
1	9.72	6.74	0.25
2	2.05	3.68	0.08
3	9.65	8.79	0.89
4	4.46	6.24	0.25
5	5.73	6.35	0.73
6	3.83	3.44	0.77
7	4.63	5.16	0.72
8	2.14	1.69	0.70

Future work

- Denny et al. believed several factors were the cause for no significance
  - may not have paid attention to additional information
  - examples not relevant to student code
- Hope to apply additional research into their messages

Outline

1 Background

2 Analyses of error messages

3 Methodologies for improving error messages

4 Conclusions

Conclusions

- Error message usability is crucial in programming
- More efforts needed on error message usability
- Research shows error messages are difficult to deal with
  - Marceau et al. found error messages do not always show underlying issue
  - Study by Traver, compiler error messages show poor usability
  - Marceau et al. recommendations in HtDP
  - Denny et al. enhanced syntax error messages were ineffective

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

References


Marceau, G., Fisler, K., and Krishnamurthi s.
Measuring the effectiveness of error messages designed for novice programmers. 2011.


Marceau, G., Fisler, K., and Krishnamurthi s.
Mind your language: On novices' interactions with error messages. 2011.


Denny, P., Luxton-Reilly, A., and Carpenter, D.
Enhancing syntax error messages appears ineffectual. 2014.


Traver, V.J.
On compiler error messages: What they say and what they mean. 2010.

See my seminar paper in "Proceedings of the Thirty-Fourth Computer Science Discipline" for additional references.

Thanks!

Thank you for your time and I hope you enjoyed the talk

Contact:

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

Questions?