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

- In programming, an error is when the computer cannot understand an expression in the code
  - these errors will return an error message
- Here's an example of an error message:

```
print("Hello World";
->java.3: error: unclosed string literal
```

Importance of error messages

- Error messages are important tool for beginner programmers
  - one of the primary interactions between the system and the user
- Unhelpful error messages impose learning difficulties, especially for new programmers
- Error messages with poor usability can lead the user down the wrong path

Goals of an error message

- An error message should:
  - not add confusion
  - be easy to understand
  - help a student locate the issue
- Example:

```
Developing...
```

Analyzing error messages

- Human-computer interaction: study on interfaces between user and programs
- Much of the research presented from an HCI perspective
- We will discuss error messages in terms of usability

Outline

Outline

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 World";
System.out.print(string.substring(6,12));

java.lang.StringIndexOutOfBoundsException:
String index out of range: 12
```

Statically typed

- All variables and/or objects assigned types
- Type checking done at compile time
  - this means different error messages
- Languages like Java or C++ are statically typed
- The following example would give an error at compile time in statically typed:

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

Dynamically typed

- Values are not assigned to types
- Type checking done at runtime
- Languages in Lisp family
- The following example would give an error at runtime in dynamically typed:

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

Outline

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

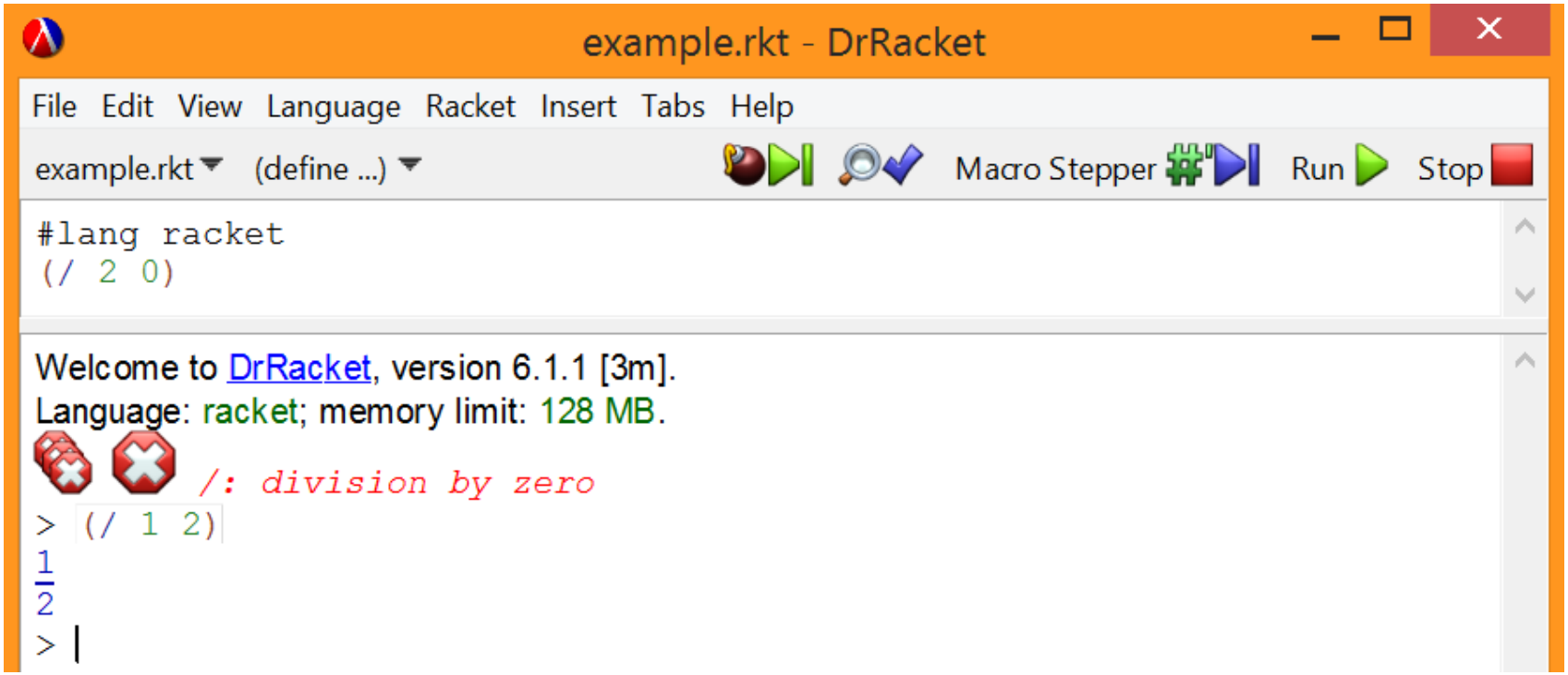
- Programming language useful for teaching in introductory courses
- Member of Lisp languages
- Functional language: computation as a composition of functions and retains immutable data and avoids changing state
- dynamically typed
- Syntax example:

```
(+ 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%	**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

KEY:
%error: Percentage of error messages during lab of the given category of errors
%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")
))
```

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::getInterestRat
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

Introduction to recommendations

- Something goes here...

Recommendations

- Something goes here...

recommendations continued

- Something goes here...

conclusions and future work for program

- Something goes here...

Java and syntax errors

- Something goes here...

How they developed the program

- Something goes here...

How they tested the program

- Something goes here...

Results of syntax enhancement

- Something goes here...

Conclusions and future work of program

- Something goes here...

Outline

Results

- Something goes here...

Future work

- Something goes here...

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.