Providing Beginners with Interactive Exploration of Error Messages in Clojure

John Walbran[1] and Elena Machkasova[1]

[1]Division of Science and Mathematics, University of Minnesota, Morris

Abstract

Programmers are imperfect, and will often make mistakes when programming and create a program error, for example, attempting to divide by zero. When a computer tries to run a program with an error, the program will halt and present the details of the error to the user in the form of an error message. These error messages are often very jargon-heavy, and are not designed to be palatable to a novice programmer. This creates significant friction for new programmers trying to learn programming languages. This work is a part of an ongoing project (called Babel) led by Elena Machkasova in an attempt to ease this friction in the Clojure programming language. Currently, Babel software is able to replace standard error messages with ones that are more helpful for a beginner audience. My contribution to this project is an exploration of potential tools to effectively display information about errors in an interactive and intuitive manner. The most promising of these tools up to this point has been Morse, created by the company Cognitect, owned by Nubank. As this project continues to explore the possibilities of Morse and how it can integrate with the existing Babel system, we are putting together potential setups that novice programmers can use to effectively understand and explore the causes of the errors they come across. This project presents the setups that have been developed and discuss their benefits and tradeoffs in helping novice programmers understand error messages. `Elena: Needs to be shortened and changed to match this year's work`

Overview of Functional Programming and Clojure

- Functional programming has a rich history of being introduced early in programming education.
- It emphasizes breaking problems into smaller, manageable, easy-to-combine pieces.
- It builds a strong foundation for students in subjects like recursion, higher-order functions, immutable data, and problem decomposition.
- Clojure is a functional that encourages these principles and is a great tool for introducing students to functional programming.
- It has a simple, minimal syntax with prefix notation in statements and statements are surrounded by parenthesis.
- It uses an interactive Read-Eval-Print-Loop (REPL) environment: the user types some code, the system evaluates it and prints the answer.

Clojure Error Messages

`Tristan: Tristan's section`

- `Elena: Examples and issues`
- Errors in Clojure are Java exceptions, as Clojure code is compiled into Java code. This means that syntax errors in Clojure will also result in an exception.
- Error messages generate when an exception occurs and provide error type, cause, and location.
- Error messages are designed for more experienced developers and may have certain terminology that does not make sense to beginner programmers.
- Example of an error message produced when the `even?` function, which expects a single input, is given two instead:

```
user=> (even?  2 3)
Execution error (ArityException) at user/eval1 (REPL:1).
Wrong number of args (2) passed to:  clojure.core/even?
```

Overview of Babel.

`Jaydon: Jaydon's section.`

- We wanted to build upon existing tools that would likely be maintained in the future.
- Maintaining a tool like Babel from completely ourselves is unfeasible for a small research group, so we instead wanted to modify existing tools.
- Previous work on this project created a tool, Babel that would provide error message

Examples

`John: John's section. Elena: (even? "two"), (even? 2 3)`

Default Clojure: `user=>(even? "two")`

```
Execution error (IllegalArgumentException) at user/eval2044 (REPL:1).
Argument must be an integer:  two
user=>(even?  2 3)       Execution error (ArityException) at user/eval2046
(REPL:1). Wrong number of args (2) passed to:  clojure.core/even?
```

Progress This Year.

- We improved internal tooling to allow us to easily select different types of error messages and test how they are presented.
- We implemented a skeleton framework for data flow to the interactive viewer.
- We created a system for labeling parts of an error message to be able to use different formats for different parts.
- We created a simple viewer to format error messages based on labeled data.

Future Work.

`Jaydon: Jaydon's section.`

Acknowledgments

We thank Morris Academic Partnership (MAP) and the Undergratuate Research Opprotunity (UROP) for sponsoring this work.
We thank Joe Lane (Nubank) for guidance on exploring this project and tools.

Sources

- [1] *clojure.org*
- [2] Morse, Nubank *https://github.com/nubank/morse*
- [3] Tao Dong, Kandarp Khandwala, *The Impact of "Cosmetic" Changes on the Usability of Error Messages*, 2019 CHI Conference on Human Factors in Computing Systems.
- [4] Matthias Felleisen, Robert Bruce Findler, Matthew Flatt, and Shriram Krishnamurthi *The Structure and Interpretation of the Computer Science Curriculum*, Journal of Functional Programming, 2004.

UNIVERSITY OF MINNESOTA
MORRIS