# A system for improving beginner-level error messages in Clojure

Ethan Uphoff, Dexter An, Elena Machkasova
Computer Science Discipline
University of Minnesota Morris
Morris, MN 2567
uphof012@morris.umn.edu,anxxx154@morris.umn.edu,elenam@morris.umn.edu

**Abstract**

Clojure is a Lisp programming language which is implemented in Java and runs on the Java Virtual Machine (JVM). It was developed and released in 2008 by Rich Hickey and has gained wide popularity around the world since then. It is in Lisp category of languages, and has simple syntax typical of Lisp languages. As a Lisp, it was specifically designed to support immutability, with an additional benefit of providing support for parallel processing. Since Clojure is very simple in its syntax and immutability-based semantics and runs on the JVM, it would be ideal to teach in an introductory level programming course. Unfortunately, Clojure has a few barriers for beginners, the most problematic one being its error messages. Since Clojure is implemented in Java, its error messages are just Java exceptions, and are incomprehensible to those new to programming. The system that we are developing replaces Clojure error message by more beginner friendly messages that avoid Java terminology.

Our system, called babel, is integrated into Clojure runtime system as middleware for the interpreter known as nREPL. It makes use of a new Clojure feature known as Spec: a system of contracts that specify requirements for function arguments (such as the number of arguments and their datatypes). Spec-based errors provide detailed information about failing arguments. However, not all errors can be processed via Spec: syntax errors or run-time errors, such as "file not found" error on file loading cannot be checked using Spec. Our system consists of several components:

1. Spec contracts that are loaded at the start of the middleware,
2. Middleware that intercepts an error message if it occurs,
3. Lookup of necessary additional information about the error in the Clojure runtime system; this lookup includes getting the stacktrace for the exeception that occurs,
4. Error message replacement
5. Optional logging which helps developers see what Clojure error messages occurred in a session and what they were replaced by.

Future work on our system will include usability testing and fine-tuning error message processing. For more information see https://github.com/Clojure-Intro-Course/babel.