Matt F, Kevin M, Nicholas N, Luis S

Dr. S.

Theory of Programming Languages

April 29, 2014

A Comparison of Programming Languages: Rust vs. Go

Our group chose two emerging languages that have seen a rise in popularity over the past couple of years. We give a comparison between Rust and Go, and see which is the better choice of language to work with. We will look at readability, writability, reliability, and cost as a rubric.

Our overview on Go

Readability: 6

Since Go is a C-like language, there are some obvious similarities to languages like C++ and Java. However, some of the things that make Go unique also makes it hard to read at a glance. The only loop structure is a for-loop, which can be modified in MANY different ways to provide various looping functions (such as the traditional while loop). A type can be declared explicitly (in the format `var <identifier> <type> = value`) or inferred (`identifier := value`), which can lead to some confusion at first.

Writability: 5

Go has some intuitive features like the `:=` operator (which is a shorthand variable declaration with inferred assignment) and multiple returns from a function. These make Go feel "natural" when writing code and it gives the programmer a sense of progress and accomplishment.

However it seems for every convenience Go offers, there are two annoyances that plague the language. For example slices are, on the surface, very simple: arrays with variable sizes. However there are many slight caveats with slices that will trip up the programmer in their

implementation (like the difference between `[0]a`, `a[0]`, and `b[0]` where `a` is a slice and `b` is an array).

Reliability: 9

Robert Griesemer, Rob Pike, and Ken Thompson wrote Go with reliability in mind. Every structure in the language is seemingly designed not to work unless used absolutely correctly. I had more trouble getting my code to compile than I did with bug fixing, simply because the language forced me to write "correct" code. For example, I used a method from the `os` packaged called `Open()` to open a file. This method actually returns two separate values: the opened file, and an error code. This actually FORCED me to implement error-handling code because otherwise, my code would not even compile!

Cost: 7

While the language may come free of cost, programmers will definitely need to be trained in the language by someone who is already proficient. Go's website provides some tutorials and documentation, but you absolutely cannot rely on just these to become fluent. Even after completing the tutorial, I had to have my nose constantly in documentation or in a forum to understand what I was doing, right or wrong. This is something that is incredibly time consuming and in a professional environment, the man-hours expense would be unjustifiable.

Grade: 27/40

We really like Go, we think it's a language that is most definitely on the right track to completely replacing C/C++ and Java in the future. We can say this with confidence because the design of the language is most definitely "smart" in that good code is almost a guarantee in the language if you can successfully compile. Yet as well designed as it may be functionally, it's incredibly upsetting to see Go suffer from readability and writability issues that absolutely

shouldn't be in modern languages. Ken Thompson saw Go as an opportunity to look at C++ and toss out everything considered outdated or absurdly obtuse, yet it feels as though in that process Go accumulated its own features that feel the same way.

Our overview on Rust

Readability: 5

Rust, like Go, is a C-like language. Rust's syntax differs wildly in certain structures, though. For example, the method header `fn method_name(parameter_name: type) -> return_type {<method body>}` is unusual and unique. Switch statements have been replaced with "match" statements which are cleaner than switch statements, but are strange.

Writability: 2

Rust's syntax is littered with special characters that make it daunting at a glance. Take, for example, the statement `println!("{:s}", "Hello World");`, which may be one of the weirdest ways to do a Hello World program that we have seen in any language. Rust is also in an incredibly alpha-like state right now. For example, there is no native implementation of List. The user has to implement their own using enums.

Reliability: 10

As with most C-like languages, Rust is strongly-typed and explicit in its syntax. Rust is built from the ground up for concurrency, which is very finicky in most other languages. Rust has built-in data "channels" which it uses to communicate between threads in a multi-threaded program, and ensure that there are no race conditions. The language is designed to be bulletproof (or it will be, once it comes out of its current nightly release state).

Cost: 6

Rust is an open-source language backed by Mozilla, which is very encouraging for its

future. However, the language in its current state is extremely lacking on resources. Even the normal Stack Overflow searches will usually fail to yield information, and when they do succeed, the information tends to be for an out-of-date Rust release. The only way to actually get up-to-date information is to crawl through the Rust mailing lists and hope you find what you need.

Grade: 23/40

Rust is a language that I feel could easily get a rating of 30 or higher in a few years. The language is currently being developed at a very rapid pace, but unfortunately, at the moment, there are too few features present, too few resources for hopeful developers to learn from, and the information out there becomes invalid far too quickly. The syntax is also confusing and needs to be improved on (however, they are actively working on this. For example, between 0.10 and 0.11 nightly, they changed `num::sqrt(some_integer)` to `some_integer.sqrt()`). I'm very hopeful about Rust, but I can't give it a good score now, in light of all of its glaring flaws.