Programming Language Go

Devon Layton
MTSU
CSCI 3210

1. INTRODUCTION

Google has an indisputable influence in modern web technology and application development; the company has been a powerhouse regarding technological advancements for years. Their rising need for a programming language that could fit the scope of more projects fueled the decision to create a language that could better fit their need. This need was met when one of Google's development teams created and released the programming language Go.

The development of Go began in late 2007 by Google developers Robert Griesemer, Rob Pike, and Ken Thompson [4]; several others joined in the development of the project over two years. On November 10, 2009, an official release of the open-source project was released to the development community [4]. Go is now generally referred to as Golang and has been a rising language in the development community since its official release in 2009 [6].

1.1 Go as a Language

Golang is heavily based on C languages, with other influences from Python and Java. Golang adapted the beneficial language constructs from its influencers yet removed pitfalls from them, thus reducing issues like slow execution or compilation, low adaptability and complex type systems. The outcome from adopting the benefits from the above languages whilst removing the pitfalls birthed the powerful language Go.

The power of Go is that the language is a statically typed, procedural language that is open-sourced [1] allowing developers to create libraries, called packages, increasing the power and flexibility of the language. One Golang's primary authors, Rob Pike, describes Golang as a statically typed, compiled language that supports garbage-collection and was originally designed to solve problems that Google encountered [8].

1.2 Golang History

Popular languages already supported many of the needs Google had but they lacked quick, readable, reusable code that Google developers needed. The Golang authors had to address the pitfalls of popular language constructs and implement features that they could not find with other languages. Constructs in pre-existing languages affected readability, writability, reliability which often required a trade-off between each other, hindering Google's ability to develop projects that were cost-effective, productive or efficient. The scope of development had drastically changed as well as both modern hardware and software, so the need for a language that addressed multiple development constructs increased drastically. Since its release as an open-sourced project, countless developers have contributed to the project which has expanded, improved, and increased its complexity.

Go was created with the mindset of taking the best features of popular, powerful languages while removing the undesired ones. Merging interpreted and dynamic constructs of modern languages into one has resulted in the efficient, reliable, and effective language that is Go.

1.3 Applications with Golang

Since Golang is an open-sourced project, a large number of applications that have been built with it are listed in the Golang/go git repository [4]. Upon visiting the repository, you can see numerous countries that have created applications with Golang.

The United States has a vast number of applications that have been built with Golang with the most noticeable being Google (the repository states much of that code is confidential) [4]. Other popular applications written with Golang include: Adobe, Bitbucket, Disney, eBay, Facebook, Imgur, Lyft, Mozilla, Netflix, PayPal.com, Twitch, Twitter, Uber [6], Zynga [4]. This shortlist of the dozens listed, reveals the popularity of the language and how much code was written to create a large portion of the applications that are used daily. Cross-platform capabilities have increased the use of companies implementing Golang and the creation of new platforms utilizing Golang [4].

2. Philosophy

Rob Pike, one of the Golang authors states "Go was designed and developed to make working in this environment more productive. Besides its better-known aspects such as built-in concurrency and garbage collection, Go's design considerations include rigorous dependency management, the adaptability of software architecture as systems grow, and robustness across the boundaries between components." [8] These design aspects have increased the popularity of the language amongst developers over the last decade because of its general-purpose use and maintainability of code once adopted into applications.

2.1 Motivations

It has been clearly outlined so far that popular languages have trade-offs between readability, writability, or reliability. The need to element as many trade-offs possible fueled Golang authors to combine the benefits of pre-existing languages whilst reducing or all-together eliminating the pitfalls in the creation of a completely new language.

Golang authors needed the ability to produce an executable quickly while also having the efficiency of an interpreted, dynamic and statically typed, compiled language [5]. The authors needed an easy-to-use, efficient language that was just as powerful as existing languages without being bogged down by language constructs.

Type systems also drove the design of the language because the goal for the authors was to minimize not only the amount of code written but also the redundancy and complexity of writing type-dependent code. This led to adopting features like type-inference similar to JavaScript and removing classes like you would have in C-languages.

The over-arching goal of the language was to remove the need for programmers to *have-to-know* how the language works but instead *do-what-you-know*. It aims to remove the Computer Science aspect of programming and replace it with just code like a programmer. This design removes the need to understand why a thing does what it does and nourishes the idea it does it what I need so it works.

2.2 Current Usage

Since Go's official release in 2009, it has reached a TIOBE language of the year twice; once in 2009 and again in 2016. It has maintained a top twenty index in TIOBE since 2016 and hit its highest position at tenth position in March 2020. The languages' popularity spiked in 2016 and has maintained a top twenty position since then.

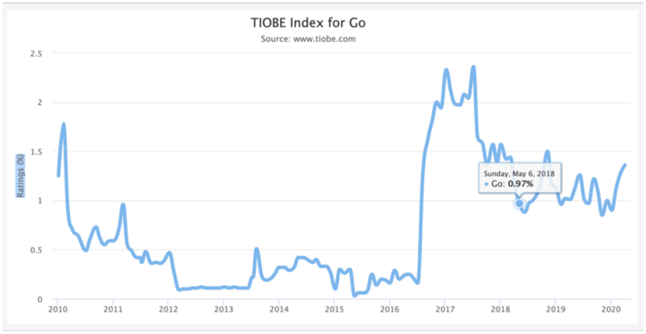

Figure 1. TIOBE Index for Go [10]

2.3 Design Issues

Developing small-scale applications with Golang produces fast, efficient executables but issues arise with large scale applications.

Issues from the design include slow build times similar to Python when coding large applications, difficult readability with code, increased production time from lack of understanding the language, and difficulty in implementing other languages with Go(cross-compatibility). Rob Pike states that "in designing Go, they tried to focus on these problems" [5] but in large-scale applications, these issues have a bigger precedence causing issues in overall development.

Golang also does not provide other language-specific elements that developers may find useful. Go does not support ternary operations, overloading, exceptions, pointer arithmetic, numeric conversions or classes; Golang also syntactically writes as an interpreted language which reduces the type errors [7].

It is also important to note that every Go program has a library that is called runtime which "implements garbage collection" and other important features so various features may not behave like it was intended to. Actual runtime checks do not behave the same way that C languages do and behavior of the program may not be what the developer intended since runtime is a library and not performed by the compiler [5].

3. Influence

Golang was heavily influenced by C languages, Java, and Python. It adopted several aspects from them while altogether creating a new, unique language with some adopted/modified features. The goal of the language was to create a dynamic language, with more efficiency which you would otherwise lose with Python and reducing the complexity that you would get from coding in C languages [5]. No classes exterminate the need for unneeded files like header files, or complex class declarations.

The purpose of Golang is to be as powerful as possible while reducing the complexity of writing code and providing the ability to create a plethora of diverse applications. "Golang is a general-purpose language with advanced features and clean syntax" [2] that provides a way to simplify writing code.

4. Criticisms

One of the biggest complaints about Go is that it has a high learning-curve. Starting out with the language can make even the most seasoned programmer feel like they do not understand the basics of programming.

Error-handling is reported without explicit user-definitions. Golang automatically returns errors in with most built-in functions, resulting in possible conflicts if the developer does not know to check for errors.

If a developer is not used to having no type system, Golang will become hard to adapt to. "Go was intended as a language for writing server programs that would be easy to maintain over time. The design concentrated on things like scalability, readability, and concurrency" [5]. Golang does not have basic types because it improves runtime and reduces the complexity of the program. The pitfalls from this is that the program may behave with undesired behaviors and result in unwanted side-effects.

4.1 The Big Criticism

Plain and simple, Go is not an easy language to grasp even for an experienced developer. The mixture between an interpreted and object-oriented design makes learning/understanding the language difficult. The caveat is the large learning curve gives rise to writing reliable and efficient code. The language allows platform independent code and powerful packages and libraries that give rise to complex programs that function on a high-level [1].

5. OOP Programming

Golang does not support object-oriented programming in the classical sense that C languages do but it has adopted and modified OOP constructs.

Golang does not support classes but gives you the flexibility to implement user-defined types and structures that mimic classes. Interfaces replace the concept of any class hierarchy you see in general OOP designs [7]. Structs are equivalent to typical OOP designs but interfaces are the main component to functioning similar to OOP. There is no type hierarchy and interfaces remove the need for the programmer to identify types because they automatically infer them; there is "no explicit relationships between types and interfaces" [5].

Interfaces are the primary way of mimicking OOP; they are the way that you define methods and members, but they simplify the approach. The trick is dynamically calling these methods and members which requires them to be methods of structs or other types that have been statically defined [5].

Golang has also removed type inheritance similar to interpreted languages. It is another step that minimizes the programmer's need to keep track of smaller details in the code. The implementation of interfaces allows types and inheritance to be automatically known and eliminates the need to manage relationships.

6. REFERENCES

[1] Anita Yadav. 2019. Go Programming Language (Introduction). (September 2019). Retrieved April 15, 2020 from https://www.geeksforgeeks.org/go-programming-language-introduction/

[2] Caleb Doxsey. 2012. An Introduction to Programming in Go: Go Resources. (2012). Retrieved March 20, 2020 from https://www.golang-book.com/books/intro

[3] Golang. 2020. golang/go. (February 2020). Retrieved March 18, 2020 from https://github.com/golang/go/wiki

[4] Golang. 2020. golang/go. (April 2020). Retrieved April 20, 2020 from https://github.com/golang/go/wiki/GoUsers

[5] Go. 2020. Frequently Asked Questions (FAQ). (2020). Retrieved April 4, 2020 from https://golang.org/doc/faq

[6] Gowiteck. 2020. Companies Using Golang. (2020). Retrieved April 19, 2020 from https://www.gowitek.com/golang/blog/companies-using-golang

[7] Keyi Zhang. The Go Programming Language Report. Retrieved March 8, 2020 from https://kuree.gitbooks.io/the-go-programming-language-report/content/6/text.html

[8] Naveen Ramanathan. 2020. Golang tutorial: Table Of Contents. (2020). Retrieved April 2, 2020 from https://golangbot.com/learn-golang-series/

[9] Rob Pike. 2012. Go at Google: Language Design in the Service of Software Engineering. (2012). Retrieved April 10, 2020 from https://talks.golang.org/2012/splash.article

[10] TIOBE. 2020. Latest news. (2020). Retrieved April 20, 2020 from https://www.tiobe.com/tiobe-index/go/