Go *does* have address the issues that make large-scale software development difficult. These issues include:

These issues include:

* slow builds
* uncontrolled dependencies
* each programmer using a different subset of the language
* poor program understanding (code hard to read, poorly documented, and so on)
* duplication of effort
* cost of updates
* version skew
* difficulty of writing automatic tools
* cross-language builds

**Semantics**

The semantics of Go statements is generally C-like. It is a compiled, statically typed, procedural language with pointers and so on. By design, it should feel familiar to programmers accustomed to languages in the C family.

**concurrency**

Concurrency is important to the modern computing environment with its multicore machines running web servers with multiple clients, what might be called the typical Google program. This kind of software is not especially well served by C++ or Java, which lack sufficient concurrency support at the language level.

Go embodies a variant of CSP with first-class channels. CSP was chosen partly due to familiarity (one of us had worked on predecessor languages that built on CSP's ideas), but also because CSP has the property that it is easy to add to a procedural programming model without profound changes to that model. That is, given a C-like language, CSP can be added to the language in a mostly orthogonal way, providing extra expressive power without constraining the language's other uses. In short, the rest of the language can remain "ordinary".

The approach is thus the composition of independently executing functions of otherwise regular procedural code.