27.2 Complexity

Restrictions of Engineering

In other engineering disciplines, we are subject to the laws of nature. Objects have limits on how fast they can move, on how dense they can be, on how much of it there is.

- Chemical engineers worry about temperature
- Material scientists worry about how brittle material is
- Civil engineers worry about the strength of concrete

However, in computer science, we've solved most of these constraints already - the sum power of Apollo missions to get us to the moon is less than the computing power of your phone.

The Power of Software

Computers have evolved over time from being large calculators to fine-tuned machines to being multi-purpose and powerful. Video games, for example, used to be customized for the limitations of operating systems but now can be built in frameworks and abstractions.

From this, the limitation is no longer the limit of computing power; it is from the ways that we plan and design what we build. Further:

- An individual programmer is no longer able to effectively manage the entire software system for a large project
  - Spotify, for example, has over a billion lines of code and 60 million used in production
- Any one programmer should only need to understand a fraction of the codebase

A Definition of Complexity

"*Anything related to the structure of a software system that makes it hard to understand and modify it*" - John Ousterhout, "A Philosophy of Software Design"

As programs have more features and functionality, their complexity increases exponentially. Consider Spotify adding a queue feature; it has to work, but it also needs to work with everything already implemented such as play/pause, search, skip, etc.

Complex systems are not a goal; our goal is to keep software simple. Complex systems:

- Take longer to understand how code works
- Are more difficult to fix bugs with confidence
- Harder to find what needs to change
  - Unknown unknowns: unclear what needs to be known to make modifications
  - Very common in large codebases

Managing Complexity

There are two kinds of complexity:

- Unavoidable (Essential) Complexity
  - To implement certain features, that feature carries some level of inherent complexity with it
- Avoidable Complexity
  - Complexity that we can address with our choices

In response to avoidable complexity, we can:

- Make code simpler and more obvious
  - Using sentinel nodes in Project 1 made life significantly easier to avoid dealing with edge cases
- Modules as a means of abstraction: the ability to use a piece without understanding how it works based on some specification
  - Interfaces are an example - HashMap, BSTMap from lab are both Maps and can be used with `get` and `put` for some key-value pairs without understanding the underlying implementation

Previous
27.1 Introduction to Software Engineering

Next
27.3 Strategic vs Tactical Programming

Last updated 7 months ago