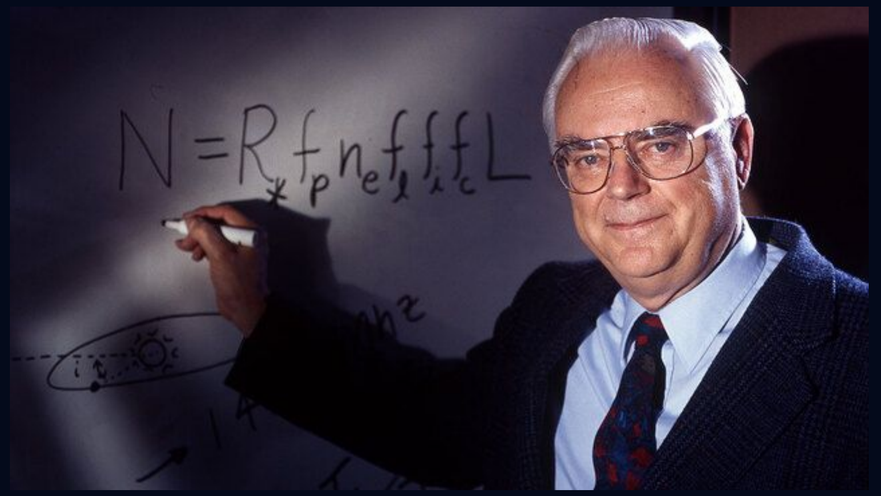
Searching for Aliens

try! Swift 2025

Noah Martin | Co-Founder, Emerge Tools

Frank Drake

- Hosted first Search for Extraterrestrial Intelligence (SETI) meeting in 1961
- How many communicative aliens are in the Milky Way?



Frank Drake and his equation

Drake Equation

Rate of star formation

$$N = \underbrace{R_* \times f_p \times n_e}_{\text{Rate of habitable planets}} \times \underbrace{f_e \times f_i \times f_c}_{\text{Fraction of planets with life}} \times L$$

Length of time they are active

N = number of planets with communicative civilizations

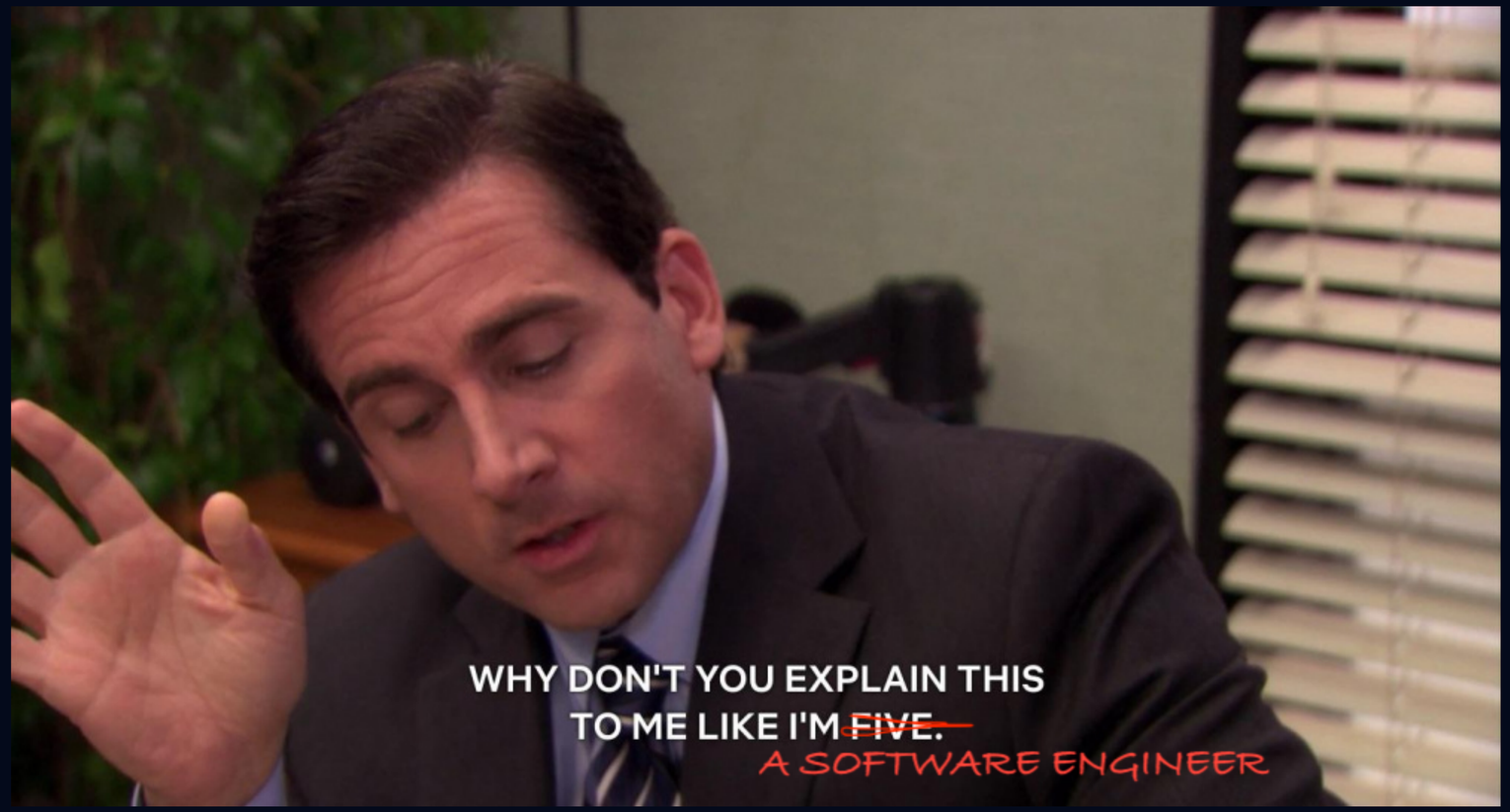
Drake Equation

Scaling to the universe

- “Archaeological version” proposed in 2016
- How many aliens have existed in the universe?
  - No need for “*L*” term
  - Replace rate of star formation with # of stars
  - Stars in the universe ~10^{22}

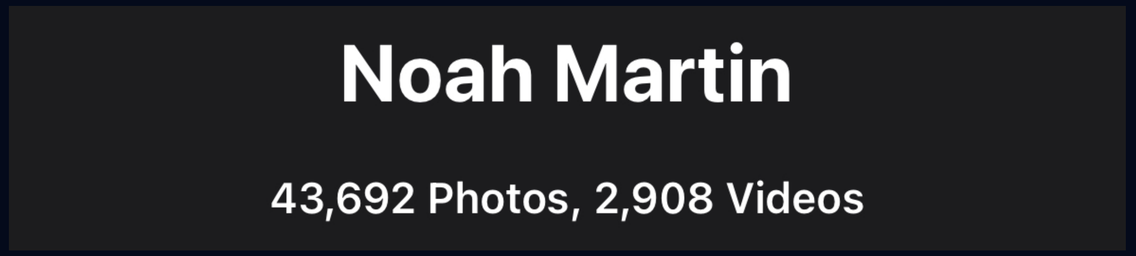
Stars in the Universe

- 10 trillion billion
- This is huge
- In the equation, it dwarfs every other term
- Even very tiny odds of life developing yields thousands of alien civilizations
  - One in a billion billion gives ~4k civilizations


WHY DON'T YOU EXPLAIN THIS
TO ME LIKE I'M ~~FIVE.~~
A SOFTWARE ENGINEER

In Software Terms

- Drake equation is linear in number of stars—O(n)
- In many cases O(n) is good!
- But imagine scrolling a list of n elements
  - How many photos in your library?



O(n) would have terrible scroll performance

Why do software engineers care?

- Look for scale factors
- Ways to amplify your work to a broader audience/greater impact
- It might be number of downloads, number of users, lines of code
- Many important metrics will mimic the Drake Equation - just a few multiplying factors

Carbon Footprint of an App

- Energy cost to download bytes to a phone
  - data centers -> routers -> phone

Efficiency of US power grid: 8.6464×10^{-5} kg CO_2/MB

App download size: 110mb

One install: 9.511×10^{-3} kg CO_2

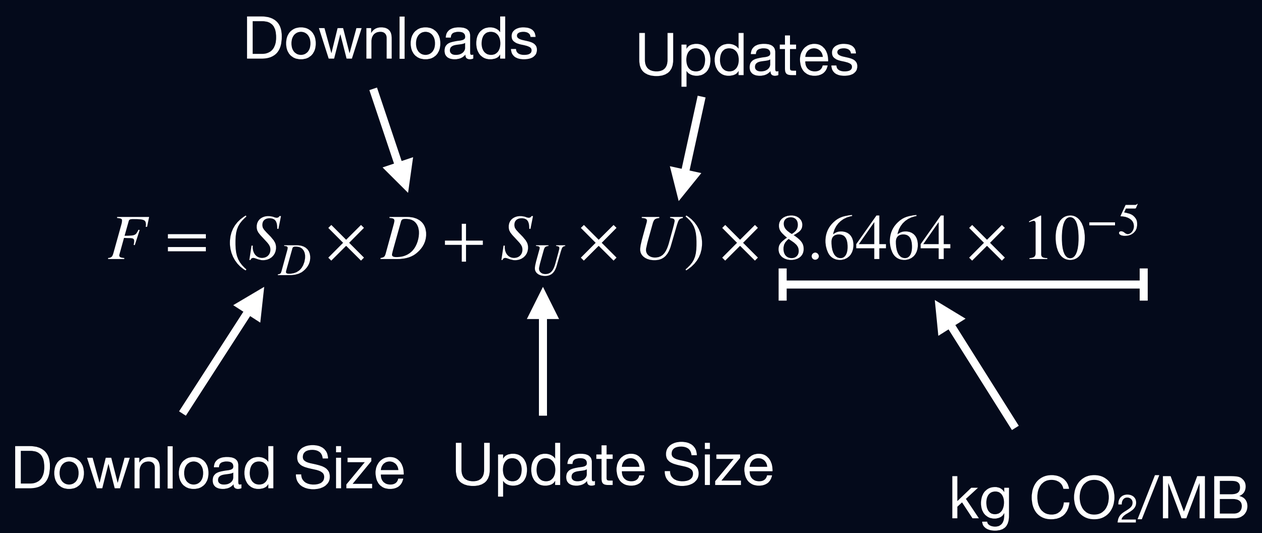
$$F = S_D \times D \times 8.6464 \times 10^{-5}$$

Download Size

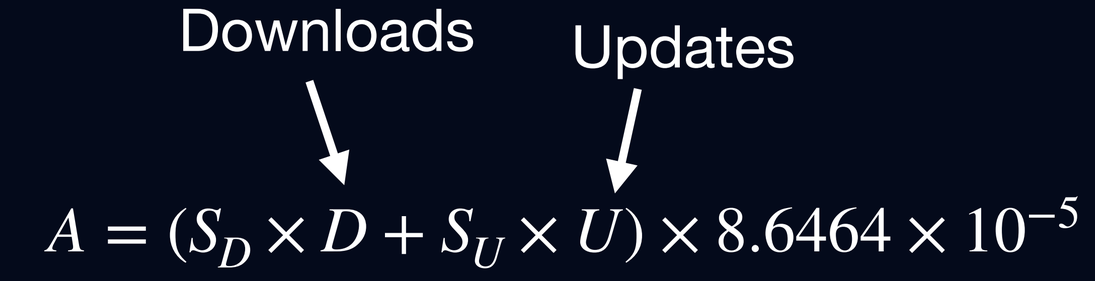
Downloads

Carbon Footprint of an App

- Apps are also updated
  - Update size is slightly smaller, but number of updates is larger



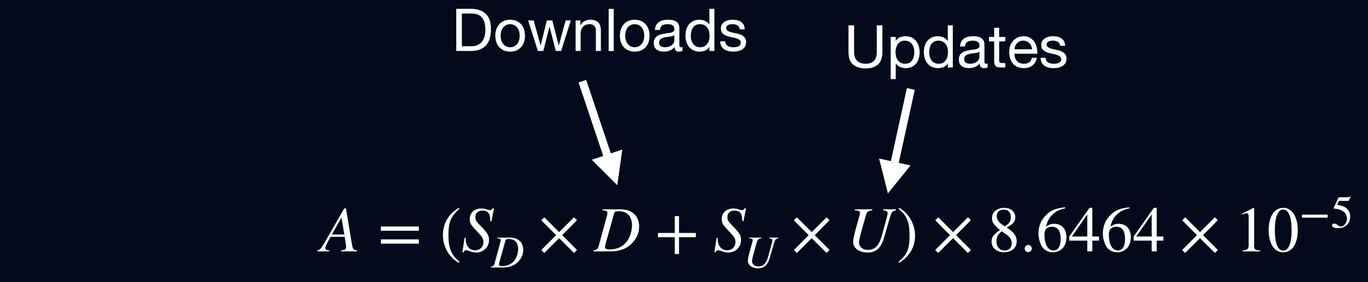
Carbon Footprint of an App



- These factors can be very large
  - Over 20 billion for Spotify in 2022[1]
- Same principle as # of stars in Drake equation
- Now a small change in app size results in a big change to carbon footprint
- 1mb can be as much as 5 round trip flights from Los Angeles to London

[1] https://engineering.atspotify.com/2023/11/the-what-why-and-how-of-mastering-app-size/

Carbon Footprint of an App



- Word of caution: Big scale factors doesn’t make it the most impactful
- Embodied carbon (building a phone) is still higher than operational carbon
- Consider ways to keep old phone functioning longer
- Performance/support for old versions

Going to Mars

Scale factors in app launch time

Going to Mars

App Launches **~10 billion times / day**

$$T_s = T_I \times L$$

Time Saved

162 days ($1.4x10^{10}$ms)

Time Improvement (1ms)

Lines of Code as Stars

- Uber iOS has a “couple of million” lines of code
- If you work with a codebase like this, there is a lot of potential!
- Each line 0.001ms faster: **1 second faster**

Lines of Code as Stars

- How an instruction is executed

With many instructions the time to read them from storage adds up

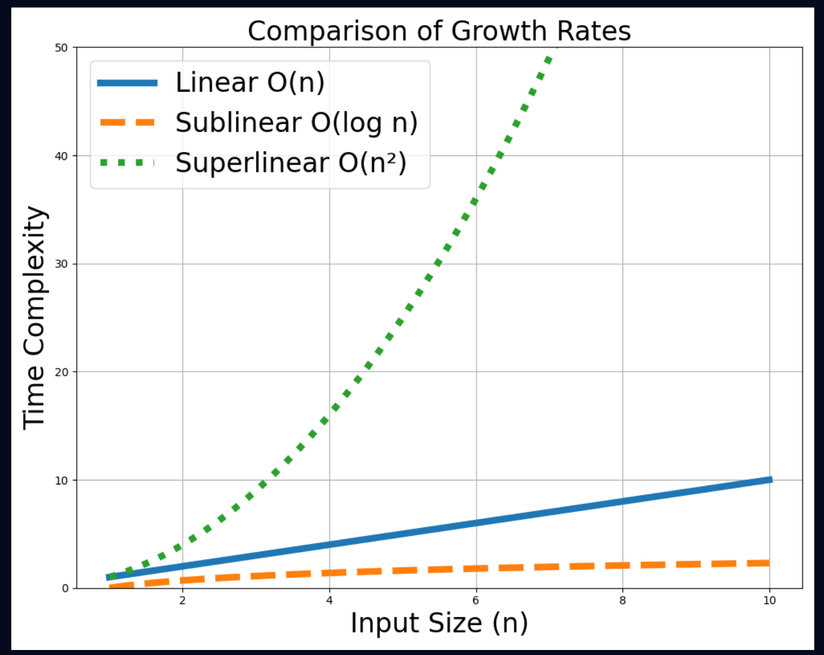
Lines of Code as Stars

Order files reduce the time
for every instruction

- Co-locates instructions that are frequently accessed
- Can load them in a batch
- Reduces the number of round trips to storage
- Have measured 18% improvements to app launch time

Non-linear scaling

- You could have sublinear scaling like O(log N)
- Less fun is superlinear like $O(N^2)$



Non-linear scaling: Protocol Conformance

- Protocol conformance check: `myVar as? MyProtocol`
- This is a runtime operation: `cast(myVar, MyProtocol.self)`
- Records of conformances stored in pairs in the binary
  - [(MyClass, Decodable), (MyStruct, Equatable), (MyStruct, Hashable)]
- Evaluating `as?` requires iterating this list -> O(n)

Non-linear scaling: Protocol Conformance

- Each check is linear in the number of conformances your app has
- It includes all the conformances in your dependencies too
- Many top apps have >100k conformances
- Is linear too slow for this?

Non-linear scaling: Protocol Conformance

- It turns out to be superlinear
- A single as? is linear
- If every protocol conformance in your app is used:

```
for conformance in conformances {
  let _ = conformance.type as? conformance.proto
}
```

Linear

Linear

- Overall $O(n^2)$

Non-linear scaling: Protocol Conformance

- Large scale factors can be performance problems
- Scaling like the Drake Equation (linear) might work
- Quadratic is worse, and that’s where issues appear
- Tips
  - Avoid too many conformances
  - Reduce dynamic casts, prefer compile-time conformance checks

The downside of scale factors

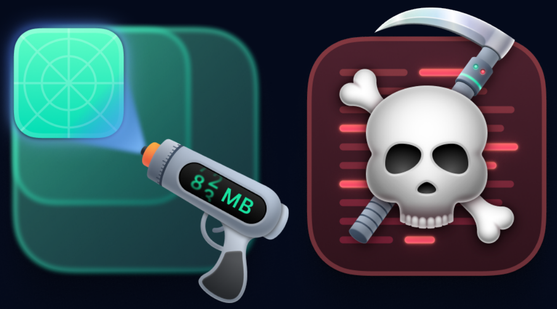
- Not always scaling something good
- How many bugs do you have per line of code?
  - 10–20 bugs per 1000 (Microsoft)
  - 0.5 bugs per 1000 (Microsoft - In release)
  - 1-25 bugs per 1000 (Industry average)
- With millions of LOC, odds are there are thousands of bugs

Write less code

- The most important change to an app is reducing the lines of code
- Drake Equation helps us realize how helpful this can be
- Good design is as little design as possible

Write less code

- The most important change to an app is reducing the lines of code
- Drake Equation helps us realize how helpful this can be
- Good **code** is as little **code** as possible
- Primary motivation behind much of what Emerge Tools creates



Write less code (continued)

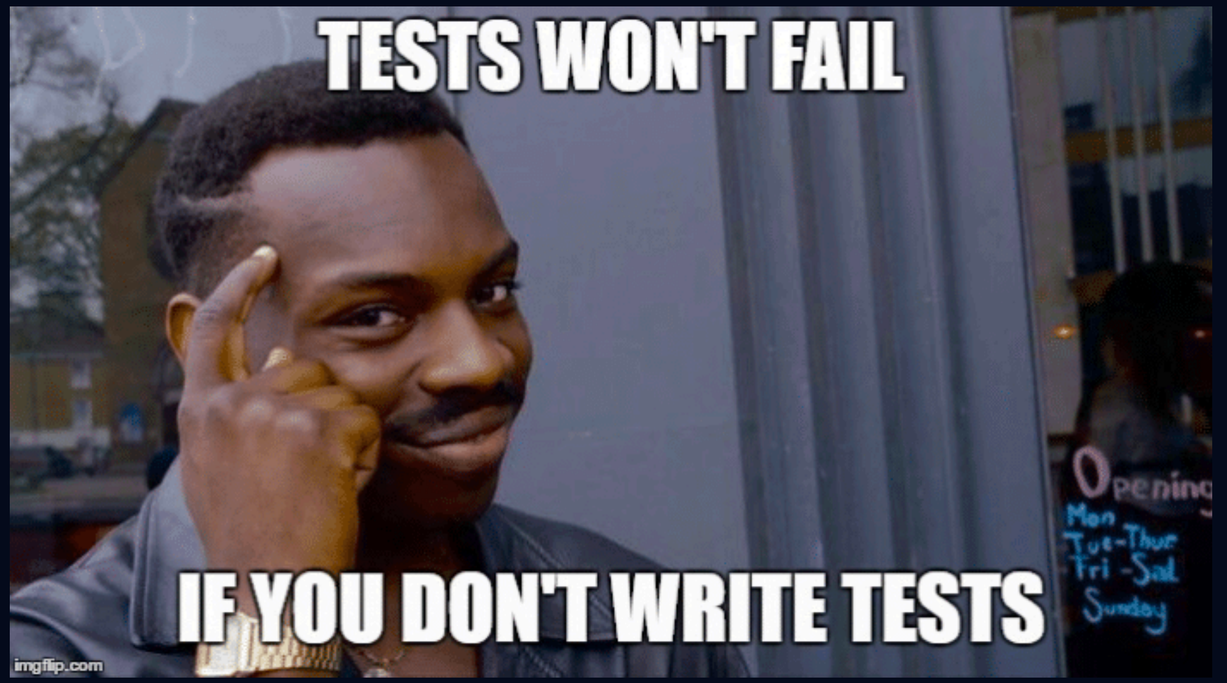
- All software is buggy!
- Codegen usually isn’t the solution
- Easy to use more code as a quick way to solve a problem, but ends up hurting more than it helps
- We do snapshot testing with no-code
  - Other methods at best codegen for each snapshot

Fundamental Principles

- Do the laws of physics prevent you from doing something?
- Higher law: App Store guidelines
- Some lessons from Drake equation are unavoidable
  - Less code is better

Test Coverage

- Is this a fundamental principle: More test coverage is better
- Writing more tests might mean more code



Code coverage in beta

- Don’t need to write a test to get test coverage
- Track used code in pre-release, what isn’t used can be deleted
  - Reaper tracks code used pre-production to give coverage of beta testing
- Automatic tests for declarative UI

Increasing your scale factors

- How to have the biggest impact?
- Look for what is used the most
  - Build systems affect all aspects of an app
- Frameworks used by many apps
- Starting a company
  - Or a conference!

Concluding Thoughts

- Think about finding aliens the next time you plan a project
- Be careful not to introduce scale factors you don’t want
- Leverage the ones that you already have
- Find me after the talk
- Get in touch @sond813

Thanks to Ken Kocienda for sparking this idea