Why does it take so long to build sofware?

Alireza Sadeghi Nasab - 1399/10/11

About the Author

simple thread

SOFTWARE DESIGN, UX

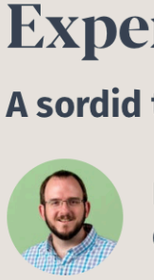
You Don't Understand Your Users

by Justin Etheredge
November 6, 2020

SOFTWARE DESIGN, USABILITY, UX

Don't Let Your User Experience Rot

A sordid tale of software dilapidation



by Justin Etheredge
October 1, 2020

AGILE, PRODUCT MANAGEMENT

Dear Agile, Can We Please Start Planning Things Again?

by Justin Etheredge
September 3, 2020

PRODUCT DESIGN, UX

Avoiding Accidental Features in Product Design

by Justin Etheredge
March 5, 2020

PRODUCT DESIGN, UX

Is Simplicity Always the Goal?

by Justin Etheredge
February 20, 2020

AGILE, SCRUM

Scrum Anti-Patterns

by Justin Etheredge
October 24, 2017

MICROSERVICES, SOFTWARE DESIGN

Gasp! You Might Not Need Microservices.

by Justin Etheredge
May 14, 2020

SOFTWARE DEVELOPMENT, THOUGHTS

Checklist for Handing Off a Software Project

A guide to making software handovers a little less painful

by Justin Etheredge
April 24, 2020

BRAND, THOUGHTS

Why Should Anyone Care About Us?

by Justin Etheredge
January 9, 2020

TECHNOLOGY

Was MongoDB Ever the Right Choice?

by Justin Etheredge
March 26, 2019

Problem Definition

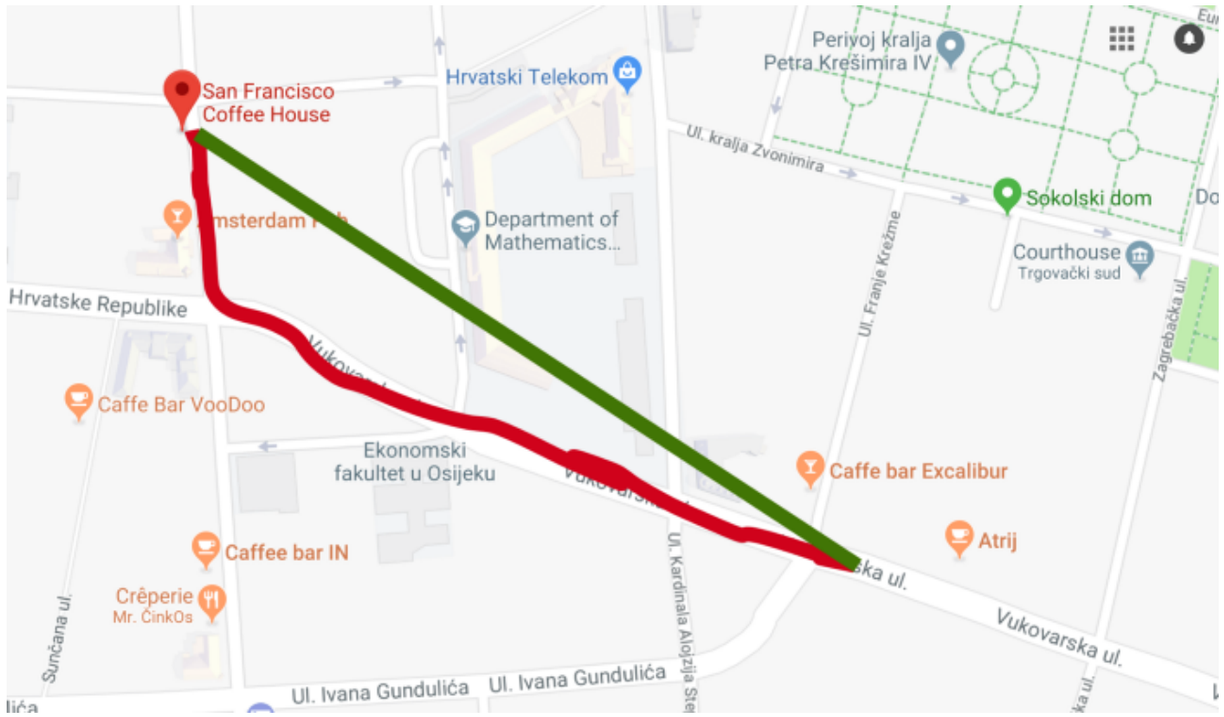
- Why does it take so long to build software?
- Why is building software so expensive?
- Why is my team delivering software so slowly?
- Why am I perpetually behind schedule with my software?

Different types of Complexity

- ⭐ Essential (Inherent) Complexity
- ⭐ Accidental Complexity

Example

☑ Buying a Coffee

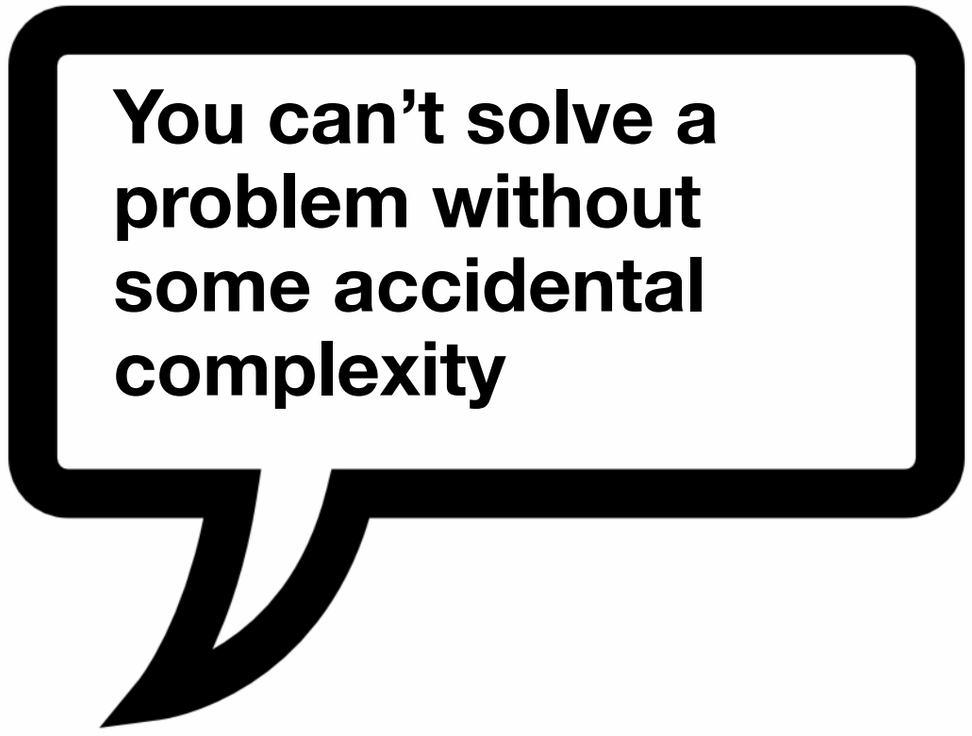


☑ Printing “Hello World!”

```
section     .text
global      _start

_start:

    mov     edx,len
    mov     ecx,msg
    mov     ebx,1
    mov     eax,4
    int     0x80

    mov     eax,1
    int     0x80

section     .data

msg     db  'Hello, world!',0xa
len     equ $ - msg
```

```
print("Hello World")
```

Words of Wisdom

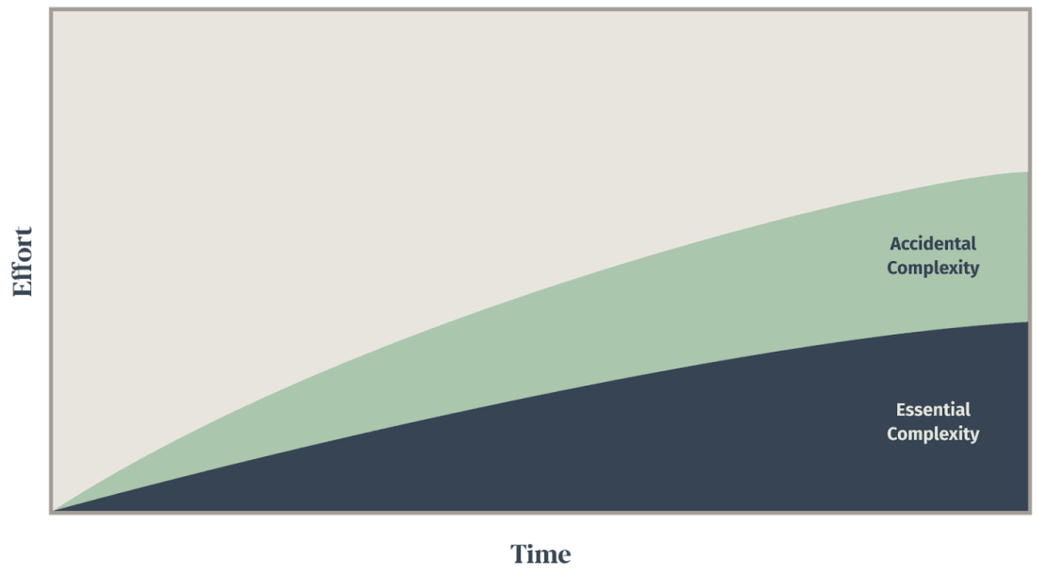


How does this apply to software?

☑ The real revolution in software over the last 20 years has been the drastic reduction in the ratio of essential to accidental complexity

☑ The Proliferation of open source frameworks and libraries has been the most powerful force for reducing the amount of accidental complexity over the last two decades

☑ The amount of code required to solve business problems versus 20 years ago has been reduced by an order of magnitude



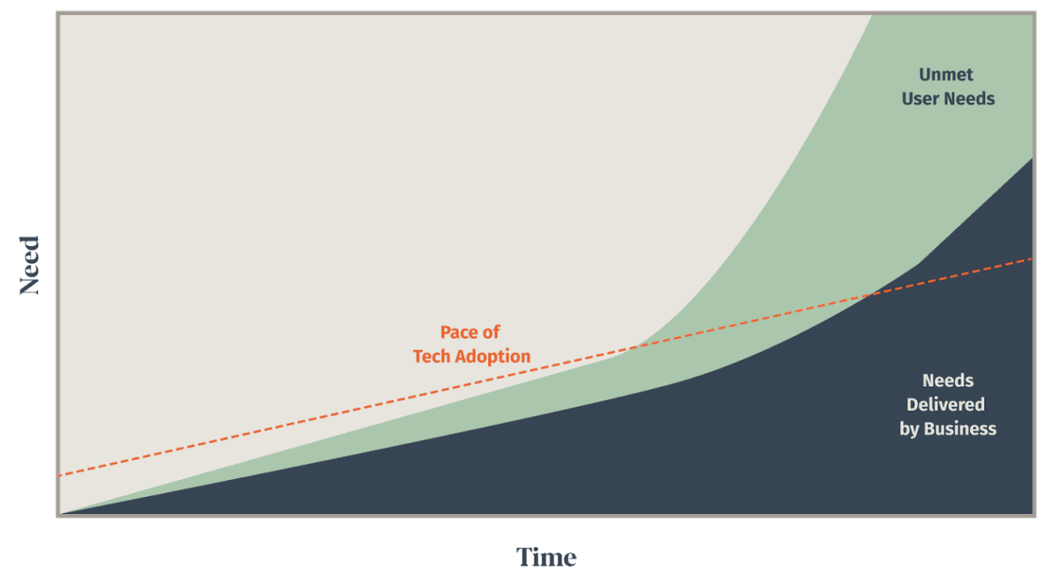
NULL

- So you would think that creating software would be an order of magnitude faster than it was back then, right?!

- Other phenomenon have been occurring concurrently:
  - We are asking more and more of our software
  - The volume of software within companies is exploding
  - The pace of new technology adoption is increasing

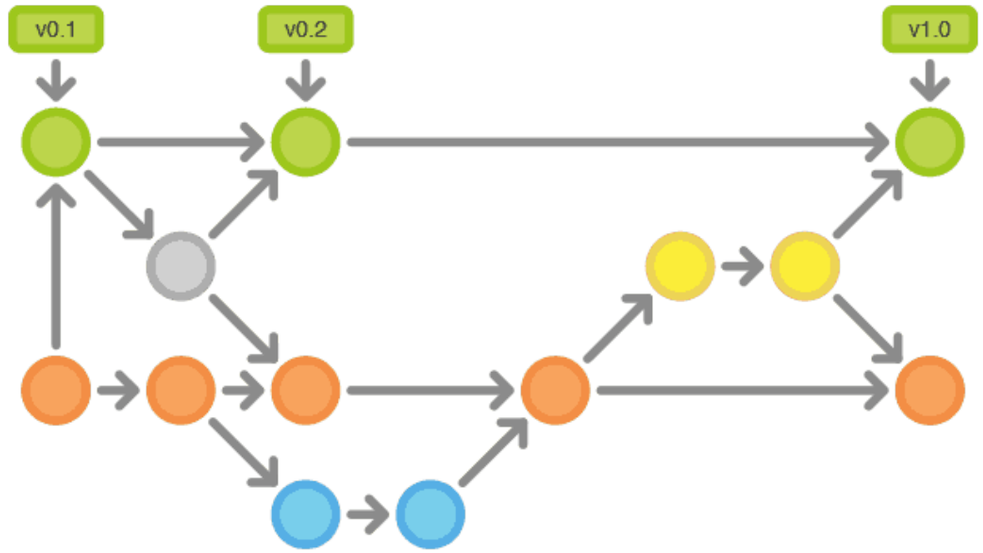
We are asking more and more of our software

- ☆ We are constantly demanding more and more from our software
- ☆ What both consumers and businesses expect from software has been increasing rapidly
- ☆ We expect software to do so much more than we did 20 years ago
- ☆ As we build these larger and more feature rich applications, in order to keep them reliable, functional, and understandable we have had to change the way we build software



Changes that we've seen across the industry

- Source control
- Automated Testing
- Splitting it up
- Specialization
- Infrastructure automation
- Frequent deployments
- Multiple devices and form factors



The volume of software within companies is exploding

- ☆ The more software that exists within a company, the more overlap between systems there is, which means that different systems need access to the same data in order to function

- ☆ As systems proliferate, and take over all aspects of business operations, they start to overlap more and more until nothing can fulfill its needs without integrating with a dozen other systems

The pace of new technology adoption is increasing

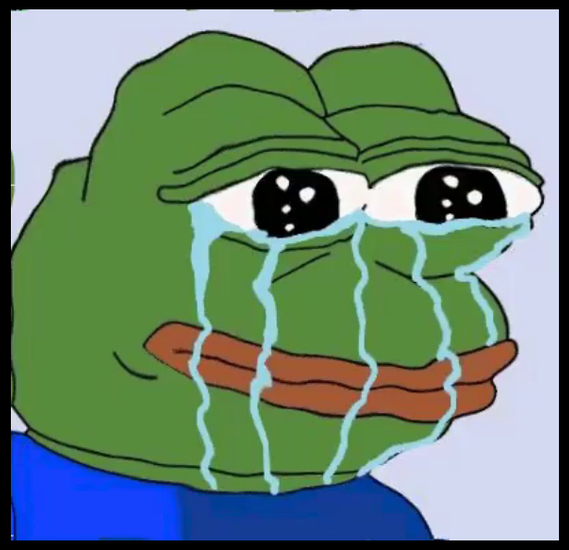
- ⭐ While using a bleeding edge tool might give you performance in some areas, the newer it is, the more you're going to feel the pain of supporting it

- ⭐ The earlier you adopt a technology, the more pain you'll experience as it grows and matures into a tool that is useful to a wide swath of users

- ⭐ Balancing the gain of leveraging a new technology with the pain that comes along with its use is something that technologists have been struggling with for a very long time

Turning off the lights!

- ★ We now find ourselves in a world where being able to sift through the avalanche of tools, frameworks, and techniques to pick out the ones that are useful (and might be around for longer than 6 months) is an incredibly valuable skill
- ★ If you’re not careful, grabbing unproven new tools or frameworks can have a detrimental effect. They can lead to a ton of accidental complexity, or even worse, a dead end if that framework dies off before crossing the chasm.



Is there hope?

- ⭐ There are certainly more reasons regarding why building software takes so long. Things such as business needs changing more rapidly, enterprise architecture standards, or an increased emphasis on security

- ⭐ The point is that what we are building in 2020 barely resembles the software we were building back in 2010, much less in 2000, and that is for the most part a good thing.

- ⭐ However, there are some downsides. It feels like we have returned to a point we were at in the 2000 to 2007 timeframe where every application was being constructed using the same tools, and many of those tools are getting progressively more complicated

Is there hope? (cont'd)

- ⭐ Many of the tools and frameworks that are now popular are coming out of large organizations that solve problems that many businesses don’t have

- ⭐ Because of this many smaller and medium businesses are finding that their ability to execute on software is diminishing rapidly and they can’t figure out how to turn it around

- ⭐ They have started to turn to low-code and no-code walled gardens in order to increase the pace of development, but in many cases they are crippling the functionality, lifespans, and ongoing maintenance costs of the systems they are building with these tools

Useful Links

[1] https://www.simplethread.com/why-does-it-take-so-long-to-build-software/

[2] http://worrydream.com/refs/Brooks-NoSilverBullet.pdf

[3] https://medium.com/background-thread/accidental-and-essential-complexity-programming-word-of-the-day-b4db4d2600d4

[4] https://a16z.com/2011/08/20/why-software-is-eating-the-world/

Thank You!