W4156

Surviving and Thriving in Software Engineering

Agenda

- ❑ Quick Intros
- ❑ Software is eating the world
- ❑ Are we prepared? (how it works in industry)
- ❑ Industry Skills and Tales
- ❑ Partnership

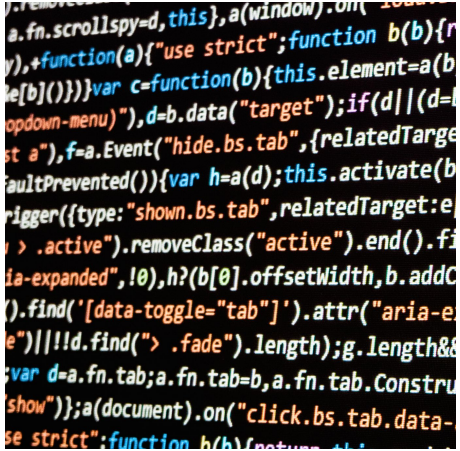
Quick Intros

Who am I and why am I here?

Quick Intros

Executive Director
Head of Architecture for Macro @ J.P. Morgan
CIB Engineering Council

Software Engineer



Unhealthy Obsession

What do I do week to week?

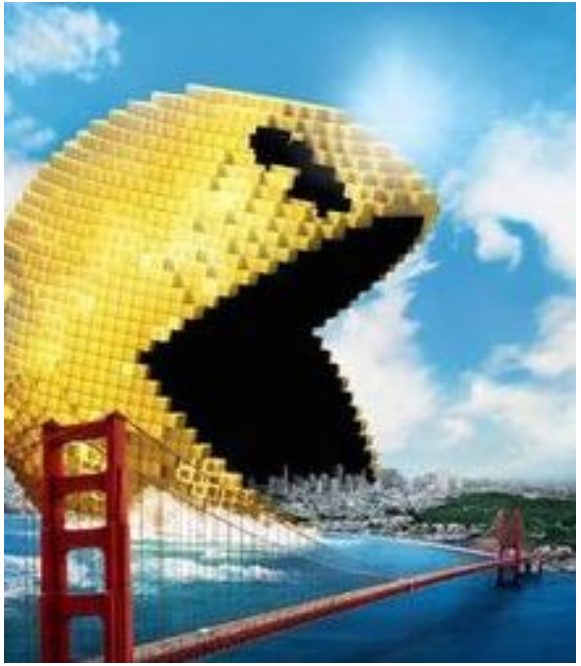
- Working with data scientists to design their next generation platform
- Product mgmt of a new customer offering
- Tech advising on investing in a company
- Pulling together 30 engineers to design a big data architecture
- Helping keep a software project on the rails
- Designing productivity improvements to our SDLC and tooling
- Building a ‘community’ for all of the engineers in my group

Software is eating the world

Why being able to ‘code’ is a near mandatory part of a future career

The importance of courses like W4156

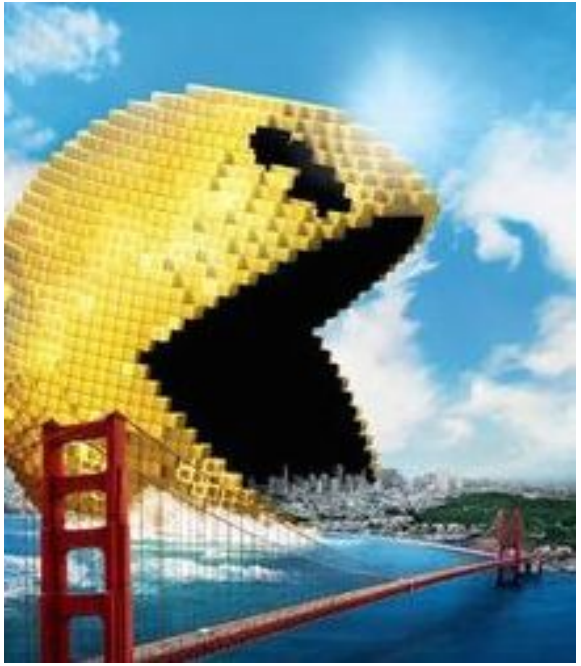
Software is Eating the World



Picture: Pixels the movie

Marc Andressen https://a16z.com/2016/08/20/why-software-is-eating-the-world/

Software is Eating the World

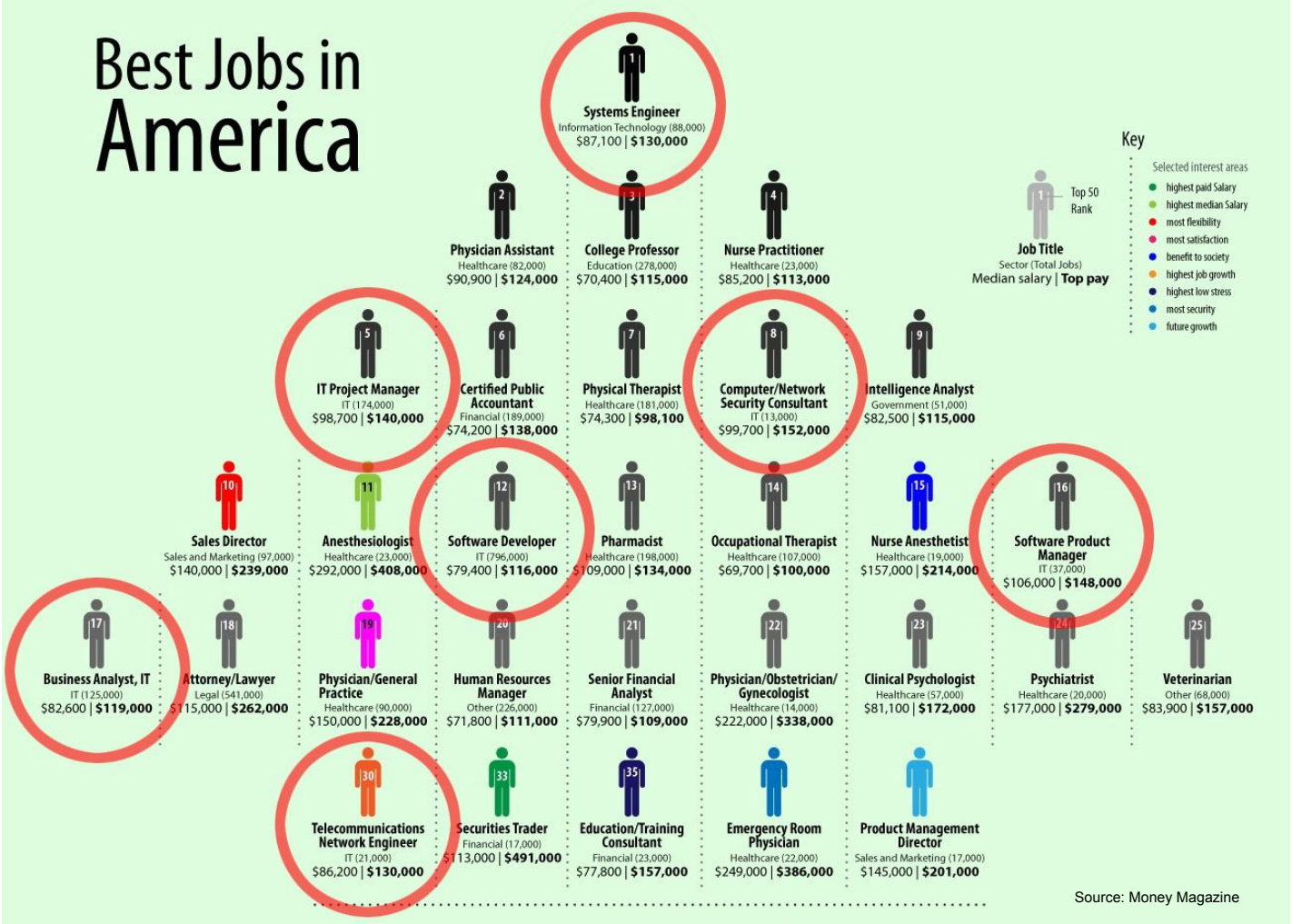


Most jobs need understanding of building software

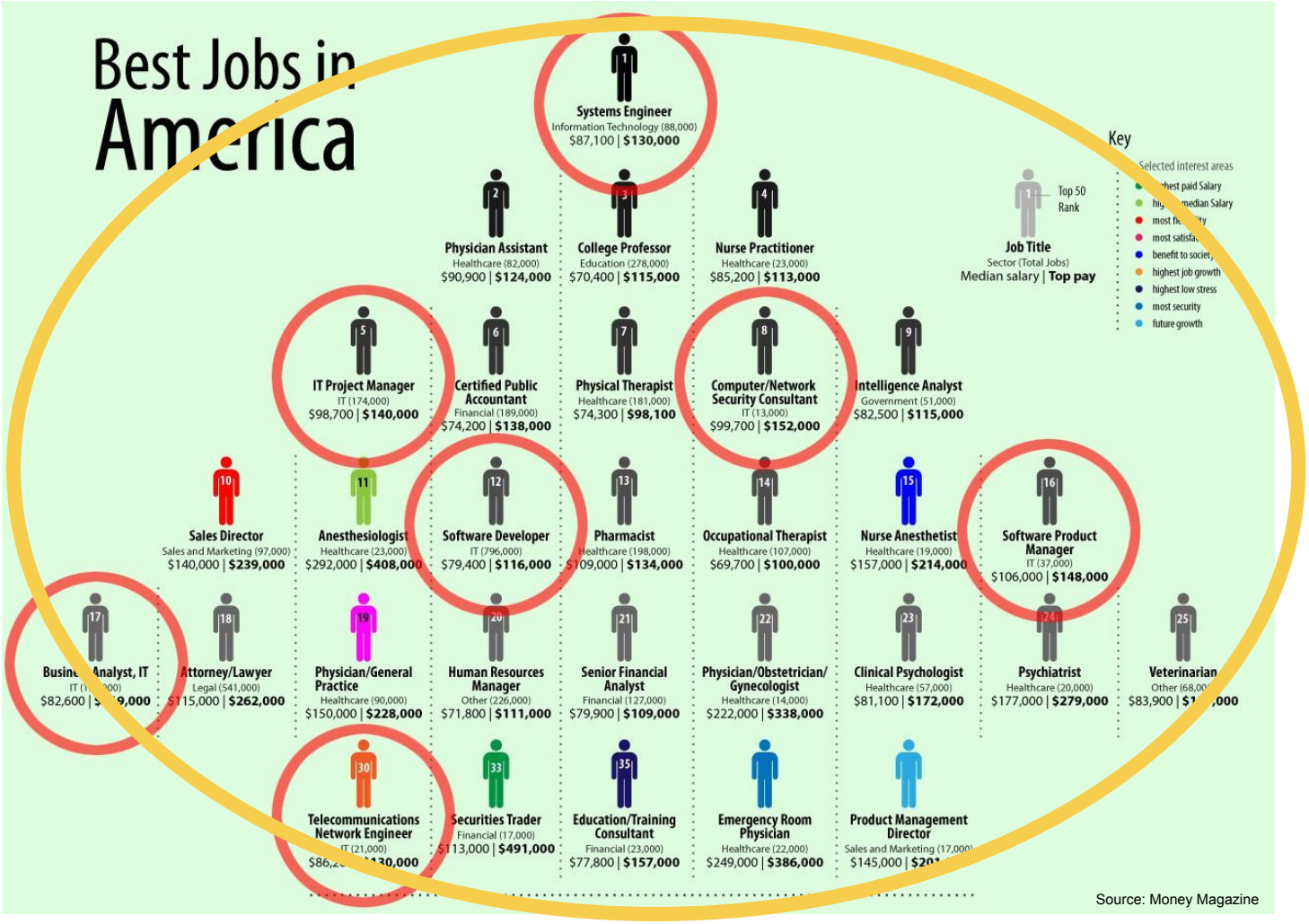
1. Software Engineering proper

2. Business users ‘self-service’
   a. analytics / MIS
   b. customization

3. Business users *asking for* software and therefore *involved* in the process



'Direct' Software Roles



Source: Money Magazine

'Direct' Software Roles

Involved in Software *Process*

BUT ..... are we prepared?

The difference between how we start to learn to ‘program’ vs how it works in industry

How we all learn to program

How we learn *‘programming’**
Well defined (provided problem / write sort etc)
Computer science ‘problem domain’
0 existing code
Don’t use existing code / plagiarism
Alone
No ‘production’
Small codebase / 4-5 weeks
No hardware failures
Focus on functional requirements

*Perfectly valid way to learn data structures, algorithms and the syntax of programming languages ……

Versus Reality / Industry

~~How we learn~~
~~Well defined (provided problem)~~
~~Computer science 'problem domain'~~
~~0 existing code~~
~~Don't use existing code / plagiarism~~
~~Alone~~
~~No 'production'~~
~~Small codebase / 4-5 weeks~~
~~No hardware failures~~
~~Focus on functional requirements~~

≠

Software Engineering in Industry
Users! Define problem. Gather requirements!?!
Hotels, Finance, ***Shellfish***, Furniture, …..
Mostly ***'brownfield'*** / existing code
Use ***anything*** that cuts time + ***integrate***
In ***teams*** (or in *organizations*)
Constantly ***running*** 24x7
M-Bn LOC to be maintained for ***years***
Real world is ***fragile***
Availability, Performance, Security, Cost, ... , ...?!?!

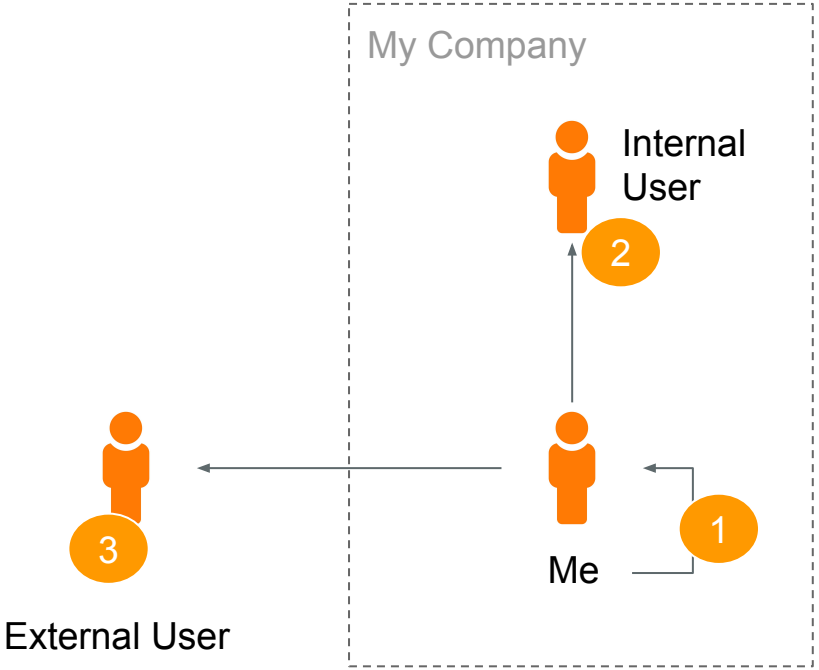
Uh Oh

Food under the door to a lone ‘rockstar’ engineer churning out code is, in most cases, *wildly* inaccurate

Industry Skills

What are the fundamentals and experience we need to thrive in industry?

Users



You will build software for all three

We need to understand:

- The problem domain
- The User(s)
- The ‘why’ and goals
- The requirements
- Which features and priority

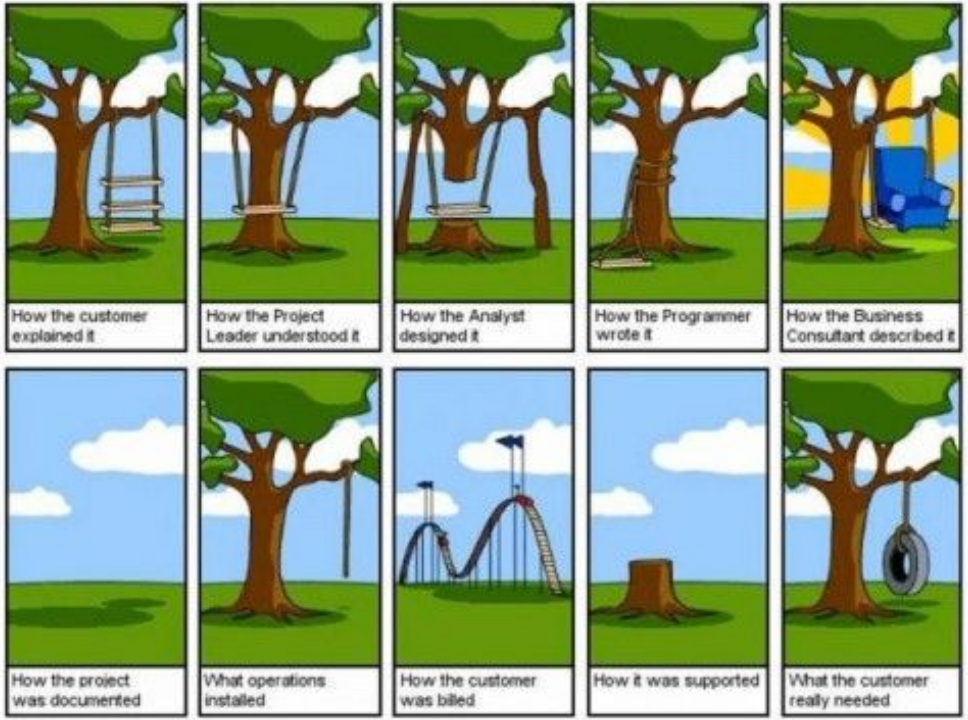
Even/especially your own startup

Product Mgmt & Requirements Engineering

"Are we building the right thing?"

versus

"Are we building it right?"



Architecture & Design

"Are we building it right?"

- Requirements to design
- "Good Code", Patterns, OOAD, DRY, SOLID,
- Technologies: Microservices, React, Swagger, DDD, Jupyter, Hadoop, Kafka, Node, CodePipeline, Blueocean, BlueGreen, Docker, Heroku, Flask, Meteor, Webpack, Pandas, ..., ..., ..., ..., ..., ..., •••

(how many technologies did I use 4 years ago and how many will I use 4 years from now?)

Testing

Majority of software that is written is not *proven* to be 100% correct

(this deeply upset me when I found this out)

‘Process’ & SDLC

We proposed all businesses are technology companies

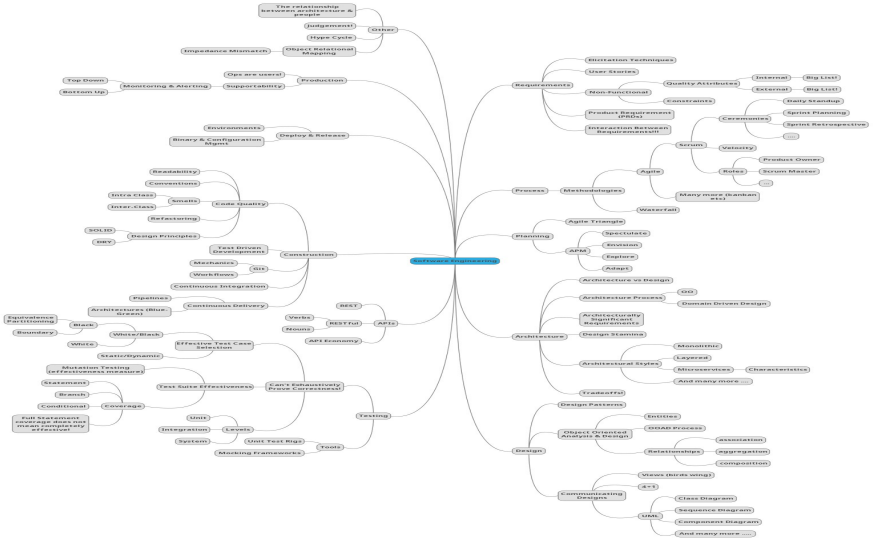
Is the agility of a business the productivity of their software engineers?

- How do we ***quickly*** deliver ***working*** software?
  (hint: the software industry is bad at this)
- Complexity
- Do 2N engineers take ½ time?

Production, DevOps and CD

Wait .... who watches my software when I am asleep?

Software Apprenticeship



Mindmap of S.E. Topics

Classmate quotes once Internships start ....

“Throughout the semester, whenever I told my best friend (who currently works as a software engineer) about the class and the topics we were covering, he always said he wished he had taken the class because such topics were already relevant to his firsthand experience in industry [...]. Now I fully understand and appreciate why.”

“[after I started my internship was] when I realized how valuable the specific material you chose to teach us was. That’s when I realized some of my peers — the ones who had previously laughed off [the lectures on] *process* rather than *coding* — didn’t understand they've never been in industry full-time. You taught us what being a software engineer in real life will be like, regardless of our naive expectations.”

“This was by far the most practical and useful class I've taken [...]. I've been helping someone on a project and am using a lot of what I learned to help the project improve (ie. building out a test suite since they currently have 0 tests in place before they deploy their product to users)”

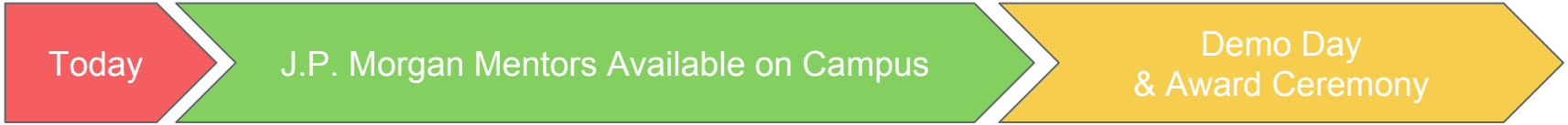
“Just a quick note to let you know that while I'm working on software projects in my internship (read: redoing one 2nd time in a week because my manager came up with a "new plan" aka didn't really think through the initial plan when handing me the project), I've come to the realization that nothing rang more true to me than the slide about the problem domain”

“I was just writing to tell you that I think I have applied a lot of the things I learnt in the Advanced Software Engineering class at my internship and it’s only been 3 weeks!”

Partnership this term

Mentorship and support from some senior software engineers from industry

Partnership Timelines



Once teams form:

- 6-7 experienced engineers available
- Be here for blocks of time on campus
- Help applying lectures to project
  - IDE, git, workflow setup
  - Product requirements
  - Engineering
  - Prioritization (Scrum phases)

(But not allowed to code the project!)

- Class comes to to 383 Madison
- Demo to JPM employees
- Judging panel of senior executives or engineers
- Award ceremony (independent of course grading)
- Drinks, Canapes and hopefully a unique view over Manhattan!

?