CS 4500
Software Development

[Intro & Overview]

Ferdinand Vesely
(based on notes by Matthias Felleisen)

September 6, 2019

Introduction

A bit about me...

Hi, I'm Ferdinand! I'm new here (started 2 days ago).
Born in what is today Slovakia, then part of Czechoslovakia.
Got my first computer at about 6 or 7 years old.

...a bit more about me...

After school, started studying philosophy, but my interests shifted back to computers
Worked as a software dev for a few years before deciding to study CS.
Moved to Swansea, Wales in UK to do a Bachelor's degree...

…and yet more about me

…stayed on for a PhD, which I did on programming language semantics and implementation.

Postdoc at Tufts, then teaching faculty in Swansea, now Northeastern.

Random: I play guitar and speak 4 (to 5-ish) languages.

Welcome to Software Development!

Purpose: scale your software development skills and attitudes to a reasonably large scale of systems building

You might also pick up some specific skills

Course

- I owe you a syllabus! – details being worked out
- Partially based on Prof. Felleisen's version
- Meeting twice a week
- Not many lectures – mainly at the beginning
    - Introducing concepts and approaches
- Rest: group presentations, design and code reviews/walkthroughs
- No required reading, but I might (strongly) suggest articles or online resources

Project and Assignments

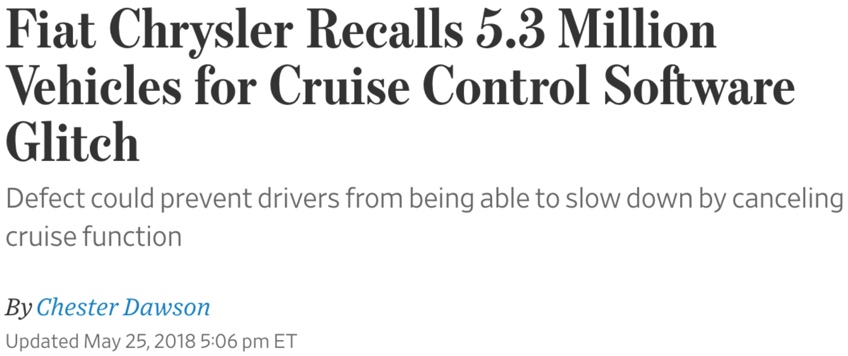
- Work in groups of 4-5
- Pair programming
- Warm-up assignments in the first few weeks, first one next week
- Grades will be based on your project and presentation of deliverables

Contact

- Nightingale 132A
- Hours: Wednesdays, 2-6pm or by appointment – caveat:
    - Depending on how busy Nightingale will get, I might try to find a different location to hold office hours
    - I might also schedule additional office hours
    - I will update you
- Email: f.vesely@northeastern.edu
- Homepage: https://vesely.io
- Details about TAs to follow

Software Development

Cost of Software (failures)


Fiat Chrysler Recalls 5.3 Million Vehicles for Cruise Control Software Glitch

Defect could prevent drivers from being able to slow down by canceling cruise function

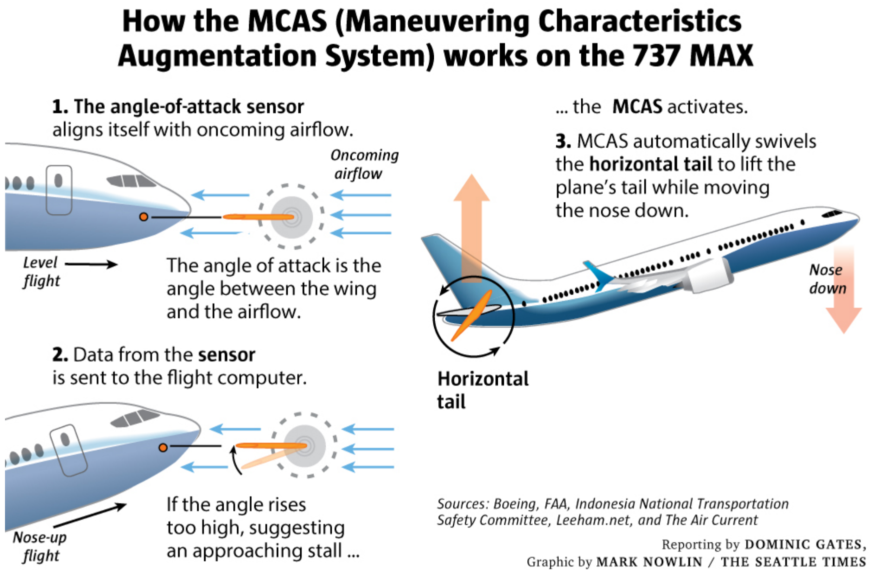
By Chester Dawson

Updated May 25, 2018 5:06 pm ET

(https://www.wsj.com/articles/fiat-chrysler-recalls-nearly-5-million-u-s-vehicles-for-cruise-control-software-glitch-1527254818)

- Problem with cruise control software
- Conditions: system accelerates, short circuit occurs
- Result: Cruise control cannot be shut off
- Even if the driver tapped the brakes

Boeing 737 MAX


How the MCAS (Maneuvering Characteristics Augmentation System) works on the 737 MAX
1. The angle-of-attack sensor aligns itself with oncoming airflow.
Oncoming airflow
Level flight
The angle of attack is the angle between the wing and the airflow.
2. Data from the **sensor** is sent to the flight computer.
Nose-up flight
If the angle rises too high, suggesting an approaching stall ...
... the MCAS activates.
3. MCAS automatically swivels the **horizontal tail** to lift the plane's tail while moving the nose down.
Nose down
Horizontal tail
Sources: Boeing, FAA, Indonesia National Transportation Safety Committee, Leeham.net, and The Air Current
Reporting by DOMINIC GATES, Graphic by MARK NOWLIN / THE SEATTLE TIMES

Figure

Boeing 737 MAX

- Software: MCAS (Maneuvering Characteristics Augmentation System)
- Triggered by erroneous angle of attack inputs
- Two sensors, but only one used to trigger MCAS – single point of failure
- MCAS would force the nose of the plane down
- Bad documentation/training: pilots could not disable MCAS
- Two planes crashed

Cost of Software (lifetime)

- Most software – prototype quality – thrown away
- Some – prototype to fully-fledged product

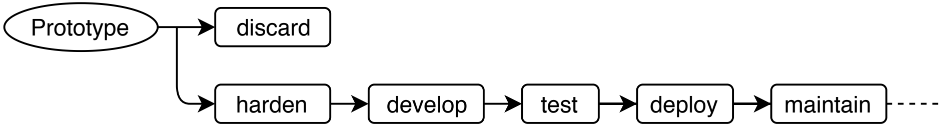
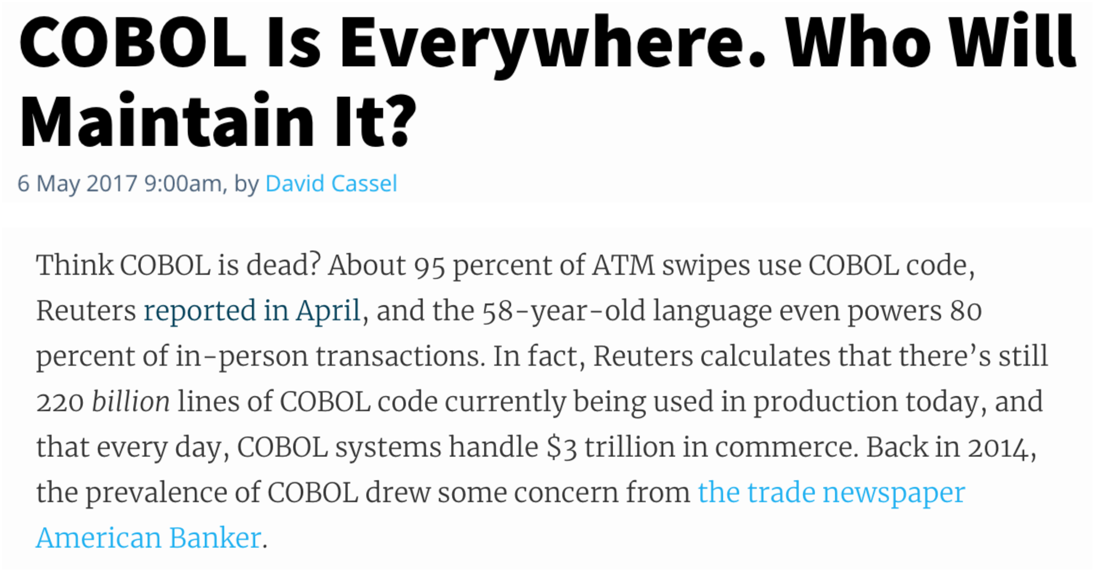


Figure: Software pipeline

Cost of Software (lifetime)

- Some code still in use today was written in the 1960s
- E.g., bank systems
  - originally written in COBOL
  - layers of Pascal, C, C++, Java, Ruby, JavaScript, PHP, …



COBOL Is Everywhere. Who Will Maintain It?

6 May 2017 9:00am, by David Cassel

Think COBOL is dead? About 95 percent of ATM swipes use COBOL code, Reuters reported in April, and the 58-year-old language even powers 80 percent of in-person transactions. In fact, Reuters calculates that there's still 220 *billion* lines of COBOL code currently being used in production today, and that every day, COBOL systems handle $3 trillion in commerce. Back in 2014, the prevalence of COBOL drew some concern from the trade newspaper American Banker.

(from https://thenewstack.io/cobol-everywhere-will-maintain/)

Where does Sw Dev come in?

"Treat your software well and it will treat you and your successors well."

- If your software survives the prototype stage,
- it will survive you (your time at the company/team/...).

Contruct your software systematically = fewer hours, less hassle, lower cost for your team/company.

Where does Sw Dev come in?

Creating software systematically

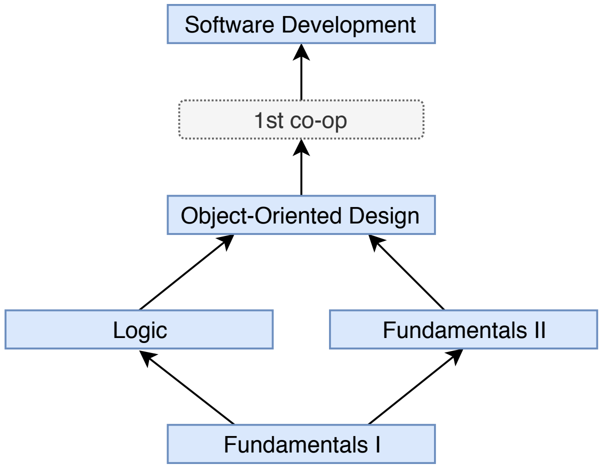


Figure: From Prof. Felleisen's *Developing Developers*

Attitude to Code

Always code as if the guy who ends up maintaining your code will be a violent psychopath who knows where you live.

Code for readability.

- Bear in mind: the future maintainer might be you!
- You never know where your code might end up

Development Skills

- **Not** a goal: to learn a particular language, IDE, tool, framework, etc.[1].->You might learn some of these along the way.
- Such skills get outdated quickly
- You will learn to pick these up faster with experience

Development Skills

We want you to develop:

- A basic idea of what “plan top-down, build bottom-up” means
- An eye for specifying interfaces and protocols
- A sense of the value of good tests and test harnesses
- Some insight into systems integration:
    - the planning that goes into it, and
    - the failures that must be accounted for

Personal Skills

From working as a group and from pair programming:

- communicating properly with a partner and team members,
- working with people with different skill sets
- working with different personalities
- coping with irresponsible partners / team members

Personal Skills

From presenting your work:

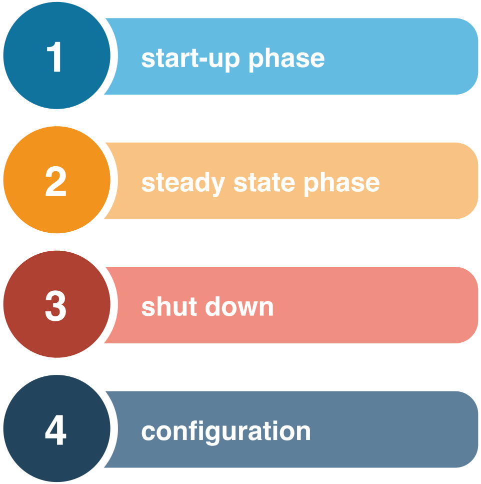
- presenting code
- accepting flaws and errors in your own code, discovered by other people

Software System

- Consists of several software components
- Clearly delineated responsibilities
- Interfaces and protocols that describe their interactions
- May consist of distributed components
- May allow extensions at run time:
    - operating systems, web browsers, web servers, . . .

Software System

For our purposes, at minimum:



Figure

Developing a Software System

Planning: **top-down**

Essentially:

1. create a “big picture view” of the system
2. figure out (rough) dependencies
3. identify runnable milestones

Developing a Software System

Construction: **bottom-up**

By analogy to building a house:

- start with the foundations
- load-carrying walls
- continue up to the roof

Next time we will take a close look at this idea.