Welcome to

Software Engineering

Thomas Fritz

Many thanks to Reid C. Holmes and Elisa Baniassad (both UBC), and Amy J. Ko (UW)

Note

This lecture is being recorded and will be offered as a podcast afterwards, **however** technical issues can occur so there is no guarantee for it/them being available. I recommend attending in person.

The videos are for personal use only.

If you don’t want to be in the picture, please sit in the back.

If there are contradictions between what’s stated in the video/lecture and the slides, please tell us/me. The slides are to be prioritized in these cases.

Agenda

1. Why Software Engineering?
2. What is Software Engineering?
3. Class Activity
4. Course Overview, Project
5. Class Activity

Introduction to Software Engineering

Software is everywhere

“Software is eating the world”

– Marc Andreesen

Medicine and Global Health

Education

Energy and Sustainability

Scientific Discovery

Security and Privacy

Transportation

Technology for Development

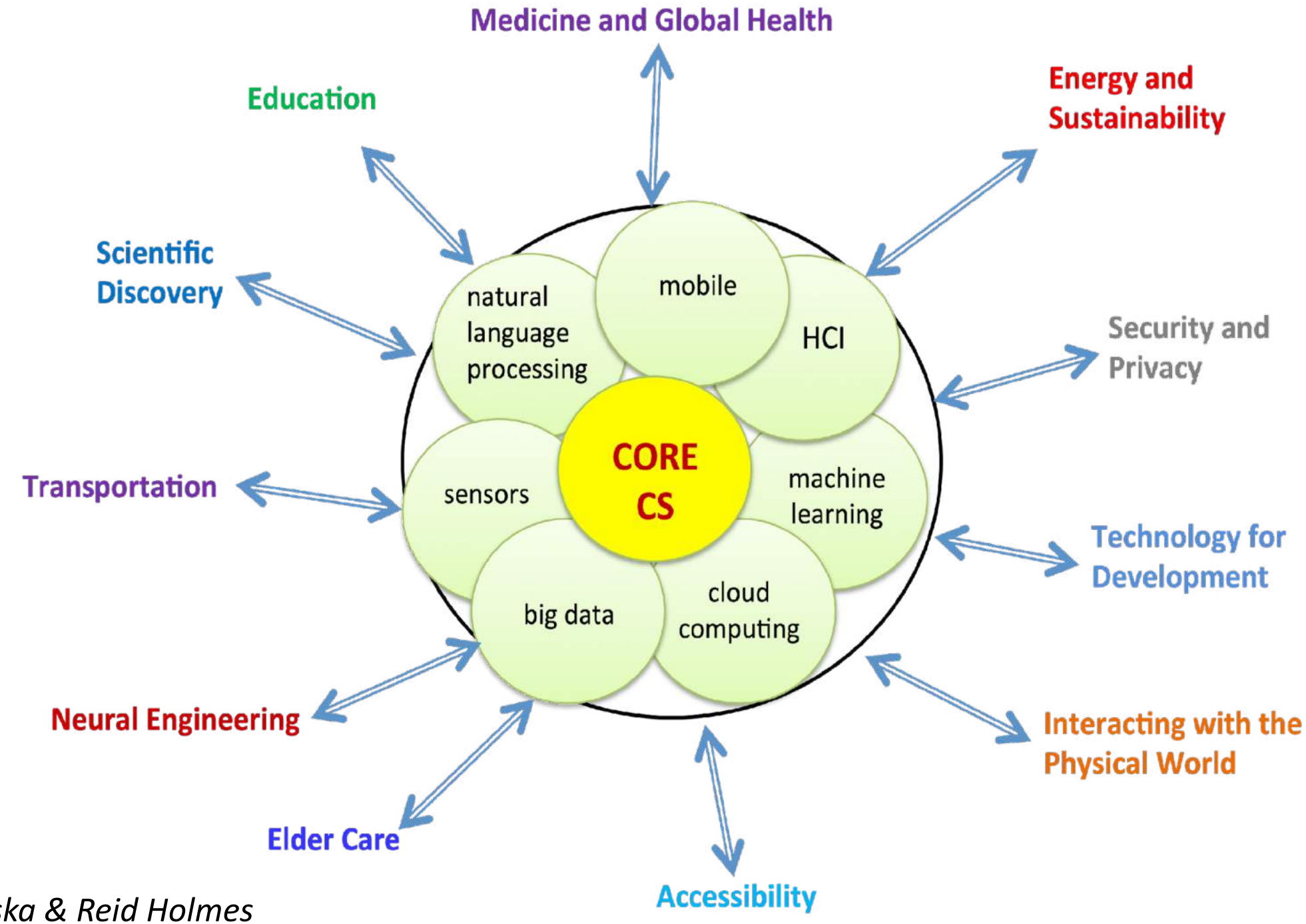
Neural Engineering

Interacting with the Physical World

Elder Care

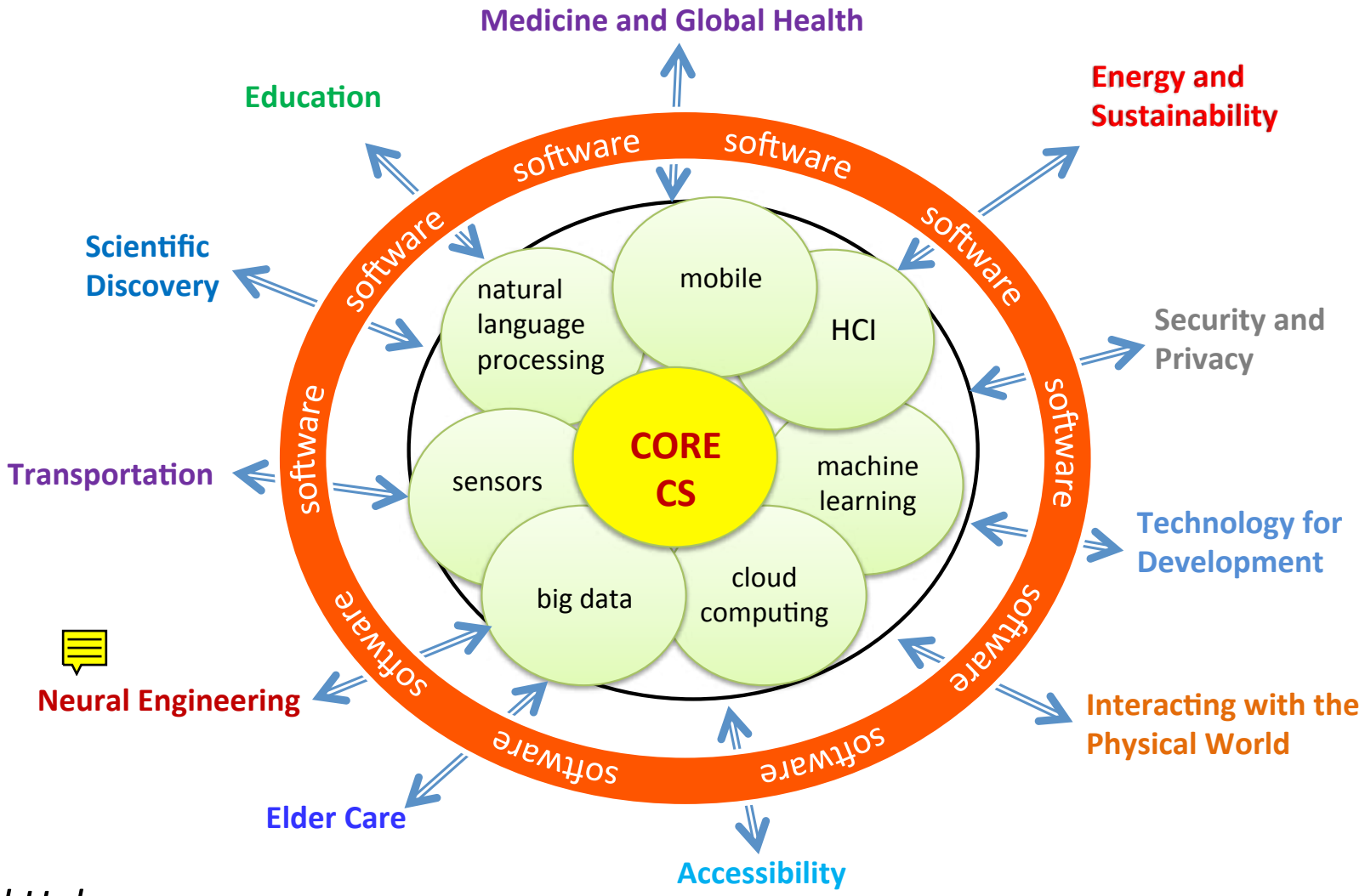
Accessibility

Ed Lazowska & Reid Holmes



Ed Lazowska & Reid Holmes

Software is everywhere



Ed Lazowska & Reid Holmes

a lot
depends
on code

What happens if it’s not engineered well?

The Ariane 5 ….

http://youtu.be/gp_D8r-2hwk

An Uncaught Exception!?

Sadly, the primary cause was found to be a **piece of software which had been retained from the previous launchers systems and which was not required during the flight of Ariane 5**.
The software was used in the Inertial Reference System (SRI) to calculate the attitude of the launcher. In Ariane 4, this software was allowed to continue functioning during the first 50 seconds of flight as it could otherwise delay launching if the countdown was halted for any other reason, this was not necessary for Ariane 5. As well, the software contained implicit assumptions about the parameters, in particular the horizontal velocity that were safe for Ariane 4 but not Ariane 5.

The failure occurred because the horizontal velocity exceeded the maximum value for a 16 bit unsigned integer when it was converted from it's signed 64 bit representation. This failure generated an exception in the code which was not caught and thus propagated up through the processor and ultimately caused the SRI to fail. The failure triggered the automatic fail-over to the backup SRI which had already failed for the same reason. This combined failure was then communicated to the main computer responsible for controlling the jets of the rocket, however, this information was misinterpreted as valid commands. As a result of the invalid commands, the engine nozzles were swung to an extreme position and the launcher was destroyed shortly afterwards.

The failure was thus entirely due to a single line of code.

Boeing 737 MAX

Photo by John McArthur on Unsplash

Malfunctioning sensor and software (MCAS) were part of the reason for 2 airplane crashes, 346 casualties

https://www.seattletimes.com/business/boeing-aerospace/what-led-to-boeings-737-max-crisis-a-qa/

“Self-Driving Tesla Was Involved in Fatal Crash, U.S. Says” (The New York Times)

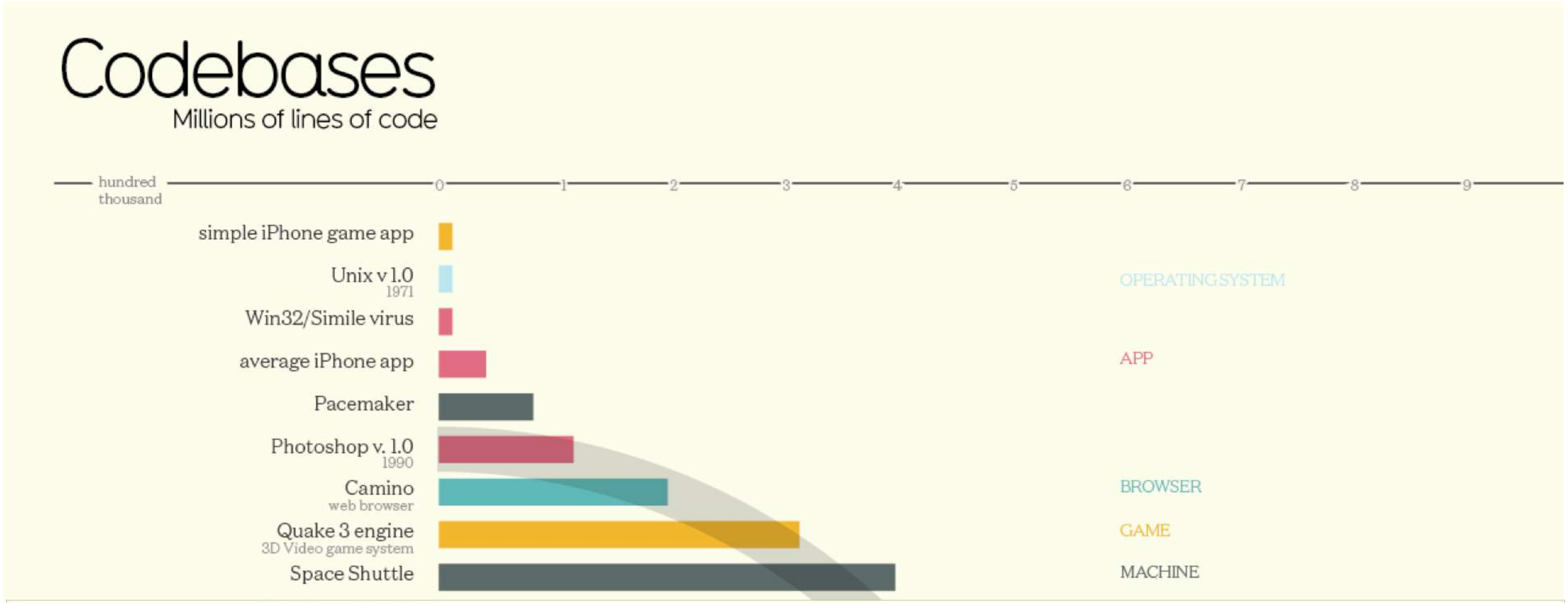
“the crash occurred when a tractor-trailer made a left turn in front of the Tesla, and the car failed to apply the brakes”

https://nyti.ms/2ktG23o

What is Programming / Software Engineering?

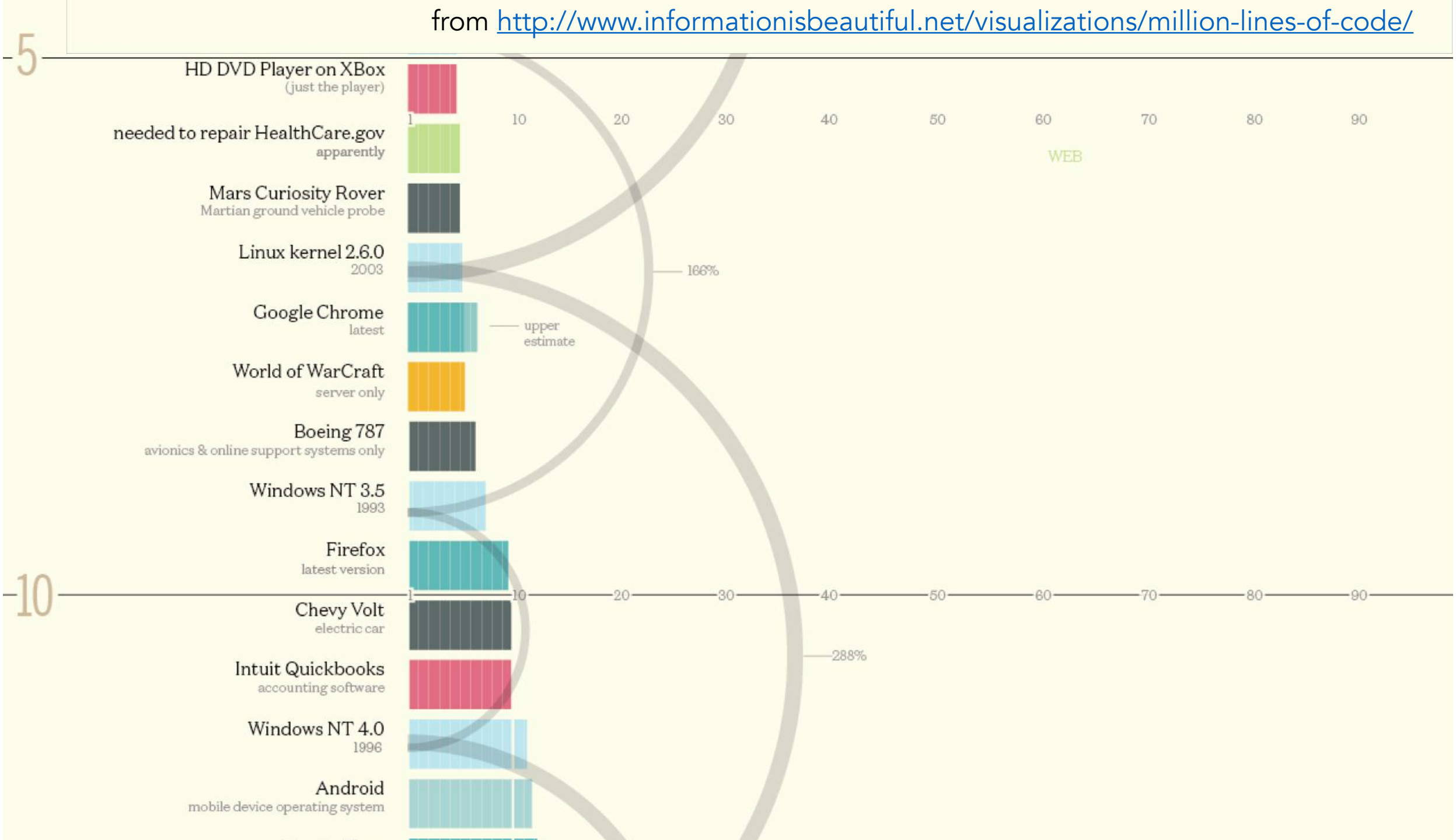
> The process of transforming a mental plan of desired actions for a computer into a representation that can be understood by the computer.
>
> — Jean-Michel Hoc and Anh Nguyen-Xuan

from http://www.informationisbeautiful.net/visualizations/million-lines-of-code/

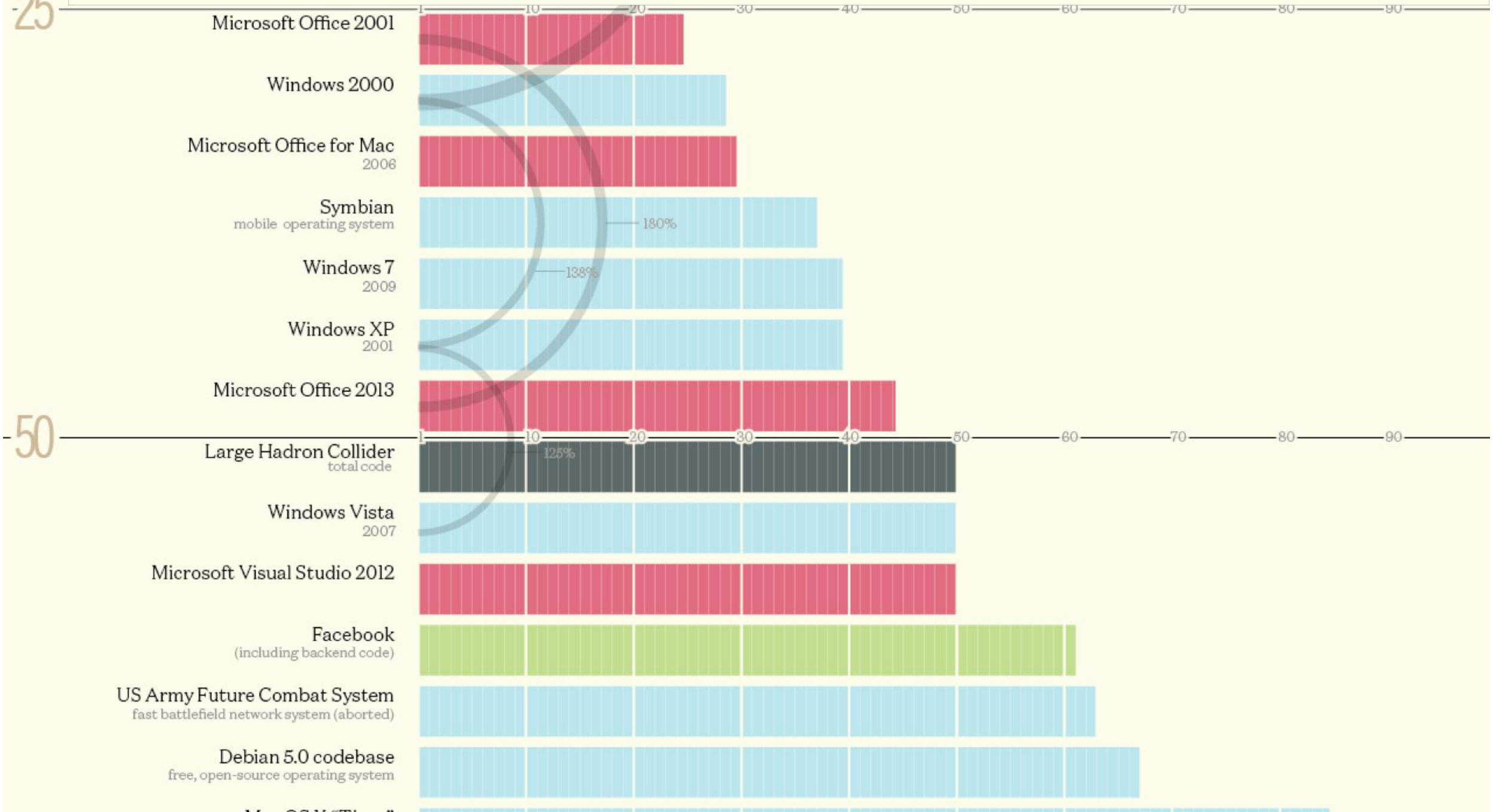


from http://www.informationisbeautiful.net/visualizations/million-lines-of-code/

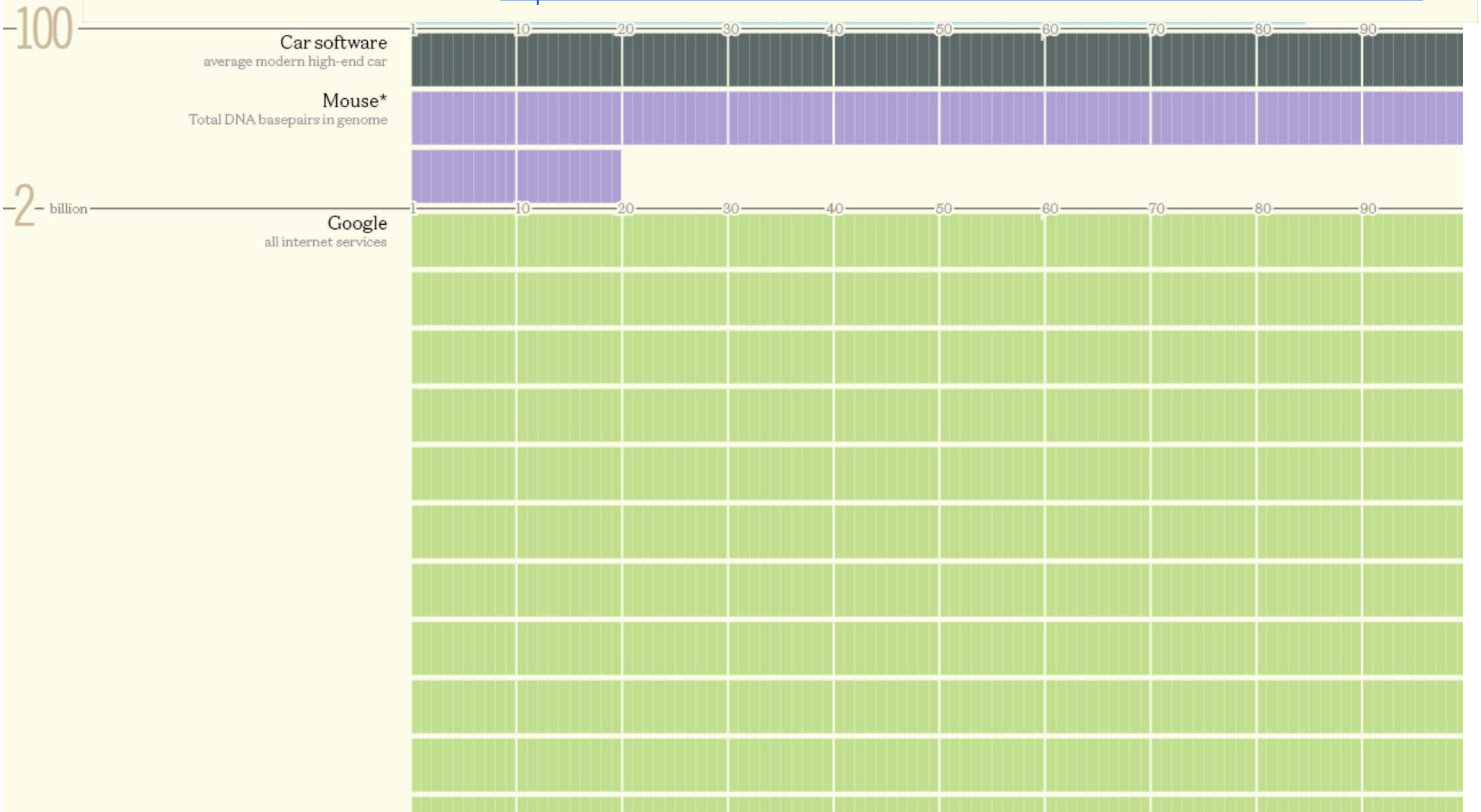
from http://www.informationisbeautiful.net/visualizations/million-lines-of-code/



from http://www.informationisbeautiful.net/visualizations/million-lines-of-code/



from http://www.informationisbeautiful.net/visualizations/million-lines-of-code/



As long as there were no machines, programming was no problem at all; when we had a few weak computers, programming became a mild problem, and now we have gigantic computers, programming has become an equally gigantic problem. —Dijkstra

What is Software Engineering?

People working **together,**
to **create**
a **robust** software system
that **satisfies the client**.

This involves **technical** and **interpersonal** challenges!

Questions for Class [Kahoot.com]

Have you ever developed software for a company?

Which SE practices did they / you use?

Why are these practices useful?

Class Activity

Engineering a system

Class Activity (groups of 4) [https://bit.ly/3T55dIO]

You and your group-mates are the C*O of a new startup called PizzaDrones, building a Pizza delivery system with drones for a big Italian restaurant chain.

For this system, spend 10 mins for the following items and fill in form:

- What are the mains steps you have to take and what are the main functions you have to develop?
- What are the main risks you have and how can they be reduced?
- What is the number and size of technical teams / developers required and the main functions of them?
- What is a reasonable timeline for full deployment of the system?

Course Overview

About me (Thomas)

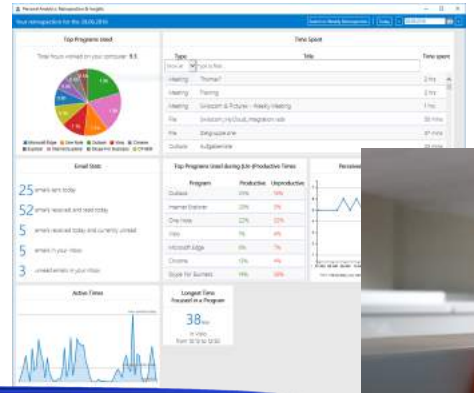
Associate Professor (since January 2018)

- (previously) Assistant Prof at UZH and University of British Columbia
- PhD at UBC in 2011 (originally from Munich)

Research area: software engineering, particularly

- Developer productivity & retrospection
- (Biometric) sensing
- Information needs & tool support

Research with companies: Logitech, CS, ABB, Microsoft, IBM, …

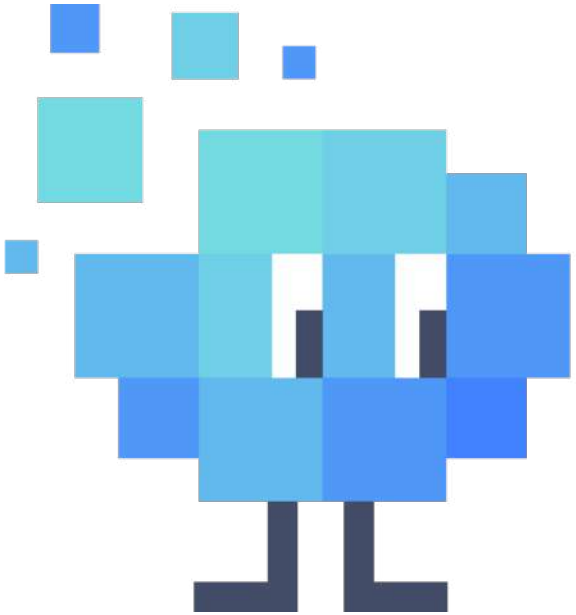


iStock

Room: 2.B.21 (usually), Email: fritz@ifi.uzh.ch

Office hours are by appointment. Email me.

Short Excurion on Research



HASEL

Human Aspects of Software Engineering Lab

productivity = ?

Industrial workers

$$Productivity = \frac{output}{input}$$

Software Developer

$$Productivity = ???$$

Photo courtesy of Christina Morillo

Rob Norris
@tpolecat

Folgen

Programming is like 5% programming and 95% random bullshit. It's gotten to the point where I feel guilty when I'm actually writing code because it seems like should be doing something more irritating.

03:52 - 11. Juli 2018

42 Retweets **142** „Gefällt mir"-Angaben

Developer Productivity & Retrospection

What does it mean for developers to be productive & well?

Survey

379 developers
28 questions

Observations

11 developers
4 hours, 2650 events

Monitoring

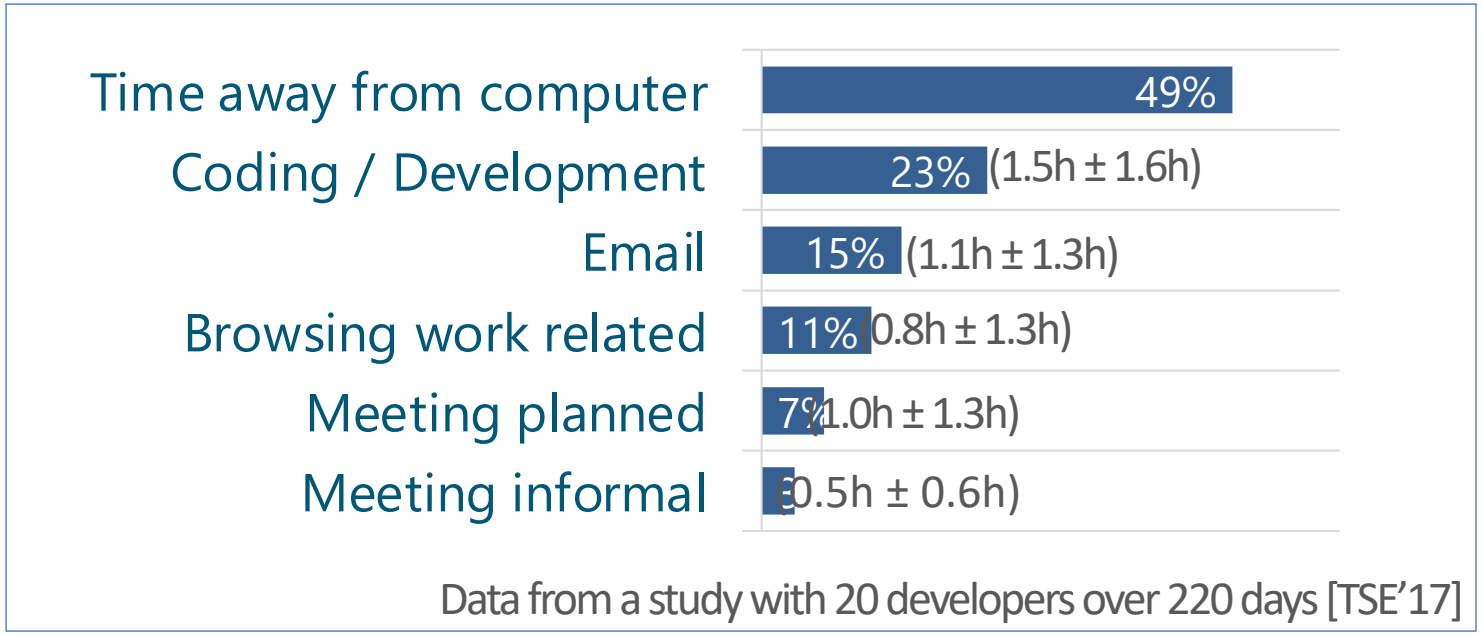
20 developers
220 days, 1350 ratings

Developers feel productive when they
make **progress on tasks** with
few expensive context switches / interruptions

Today's Work Life of Software Developers

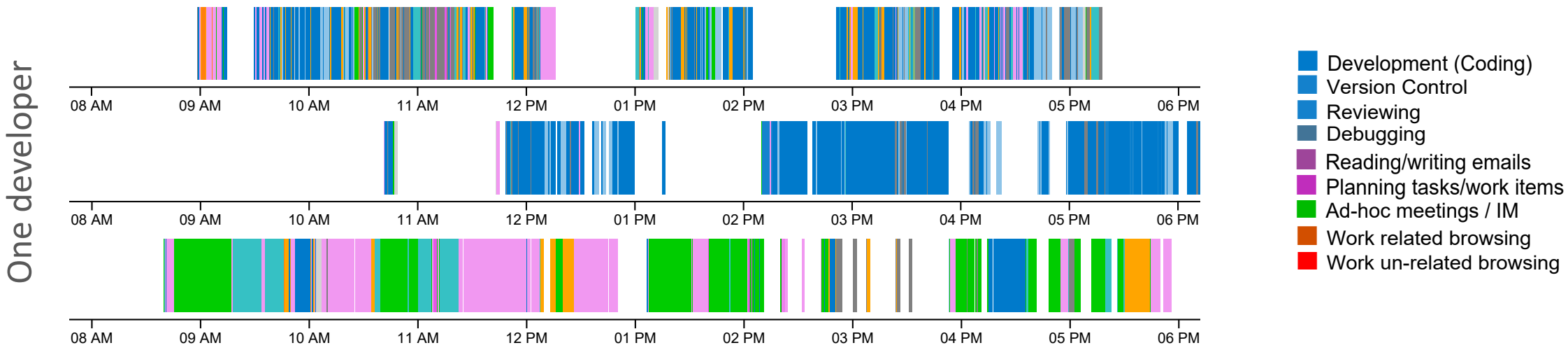
Multi-facetted work with a broad variety of complex tasks

e.g. implementing new features, fixing bugs, reviewing code, testing, building and deploying changes, meetings,…



Monitoring – developers' work & productivity

20 developers, 220 days, 1350 productivity ratings



Development work is highly fragmented

- 0.3 to 2 min per activity

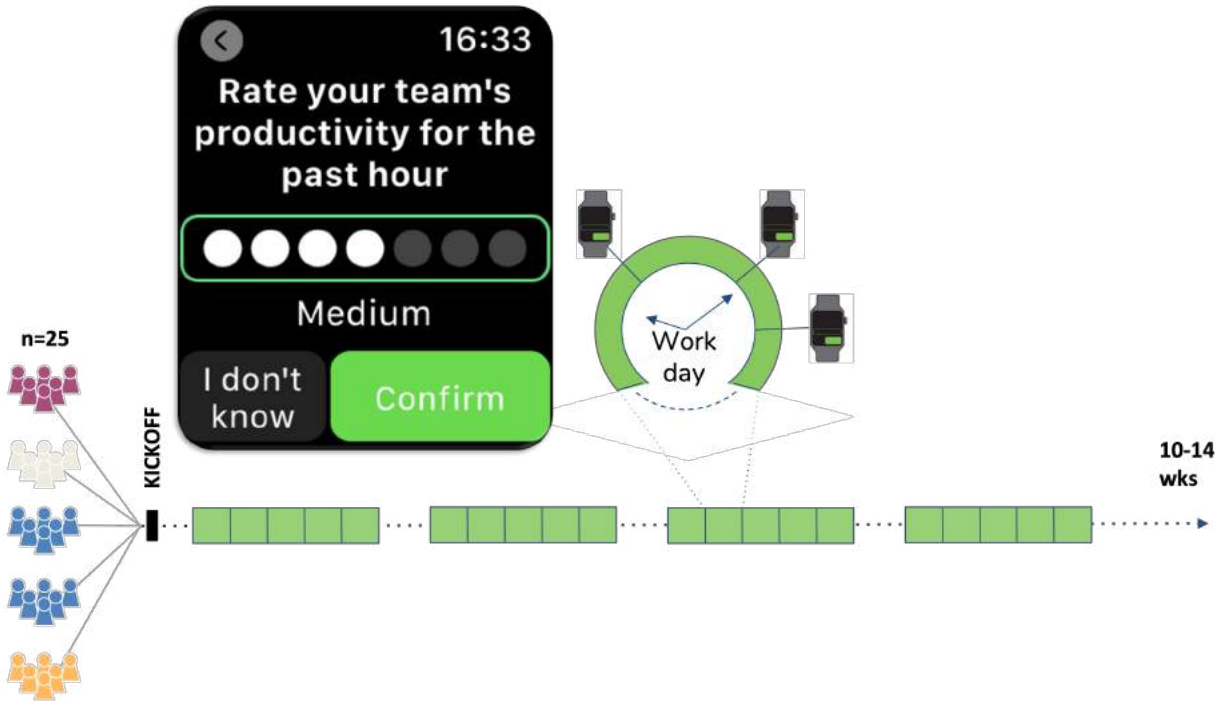
Productivity varies a lot by individual, ToD, activity,…

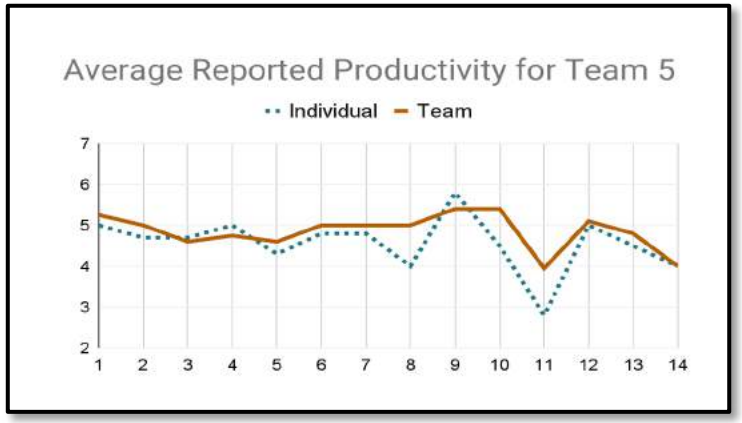
individual productivity ≈ [progress on tasks - cost of context switches]


Balancing Individual
and Team Work

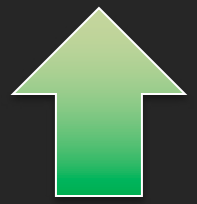
Perceptions of Productivity in Software Teams

Longitudinal study (10-14 weeks)
25 developers, 5 teams



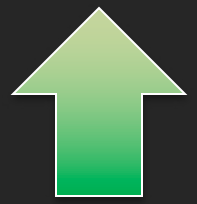


- Developers perceive their own productivity very closely to their team's, they are **interrelated**.
- Software **teams are fluid**, having multi-level structures and changing frequently
- Managers view productivity differently than individual contributors: they see the importance of **project progress as a whole** rather than emphasizing individual productivity.



team productivity

$\sim$



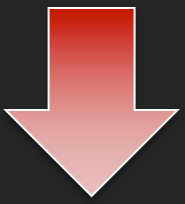
individual productivity

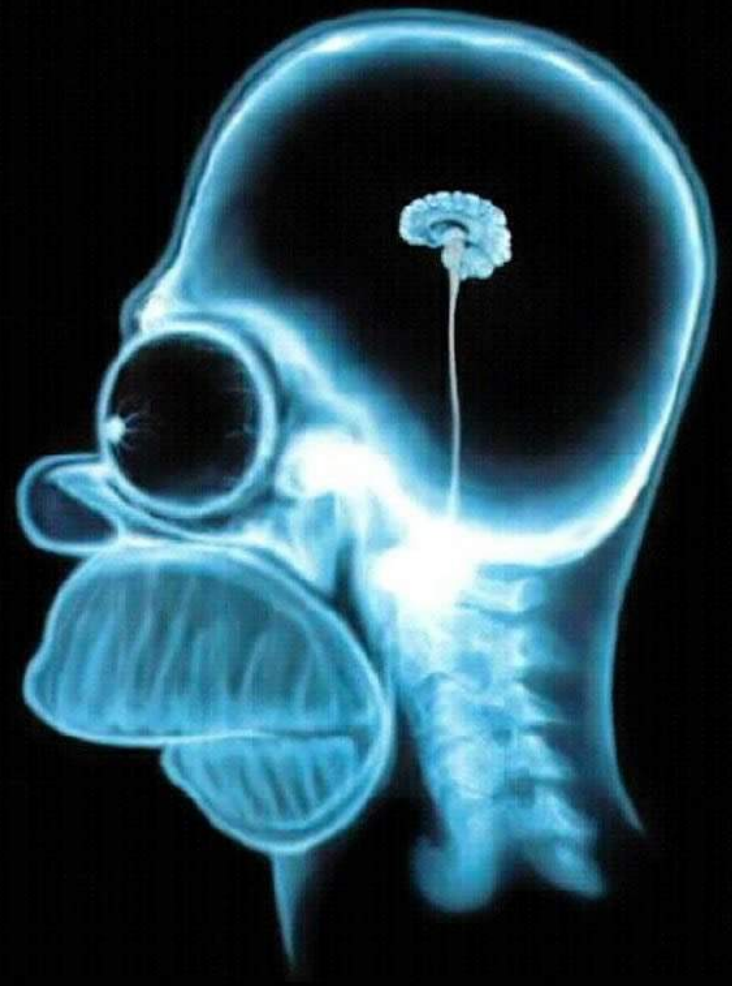
$\sim$

progress on tasks

-

cost of context switches





Biometric Sensing

40

FlowLight

PULSATING RED

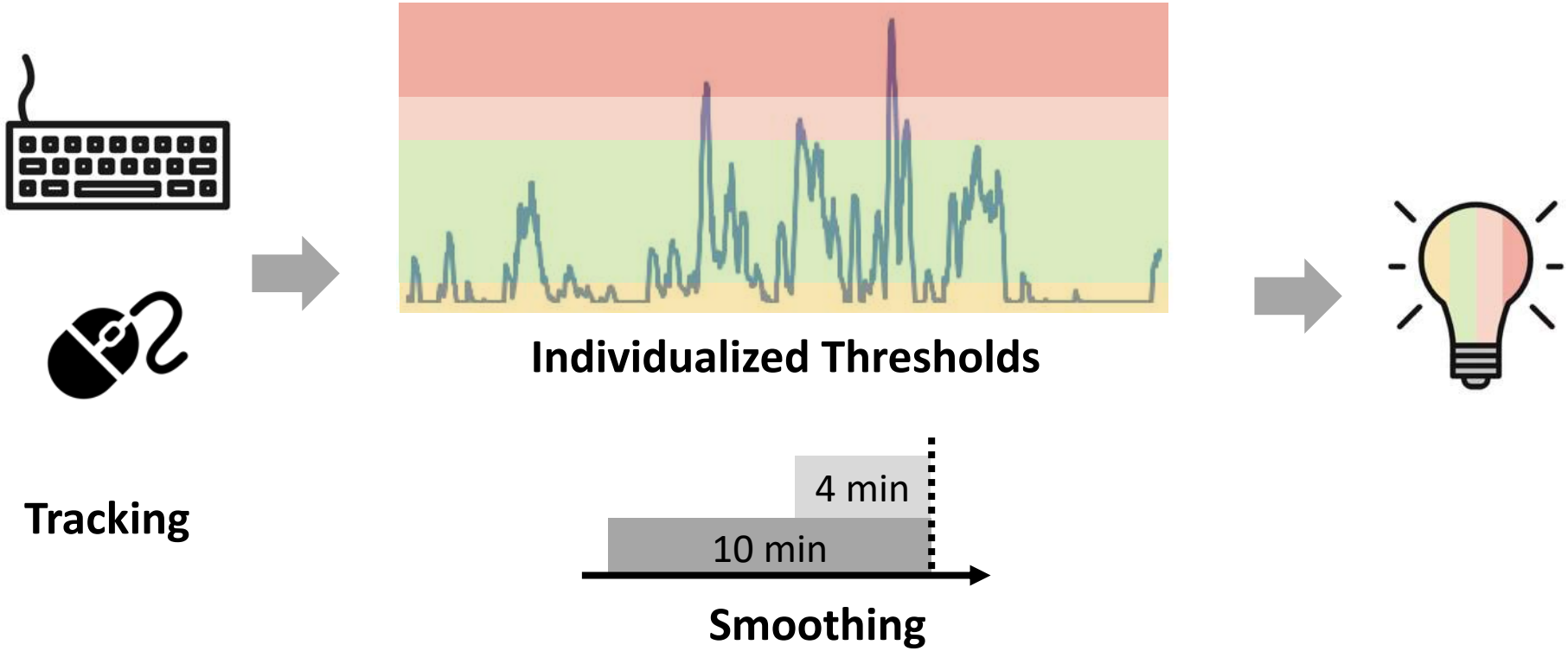
RED

GREEN

YELLOW

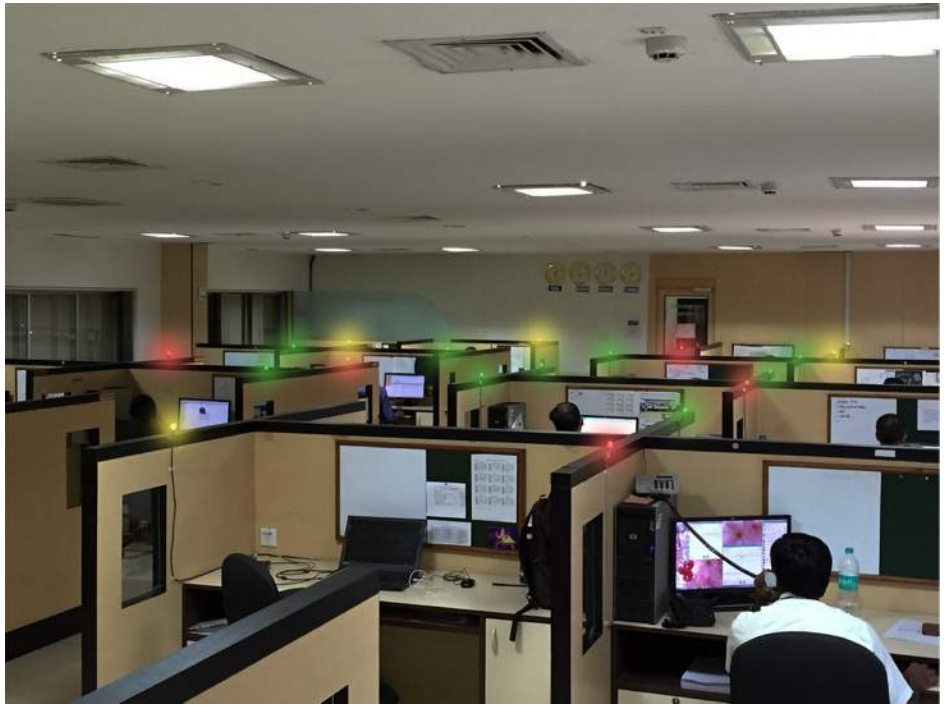
FlowLight – automatic interruptibility sensing

Field study with 449 participants, 12 countries



FlowLight – reducing costly interruptions

Field study with 449 participants, 12 countries



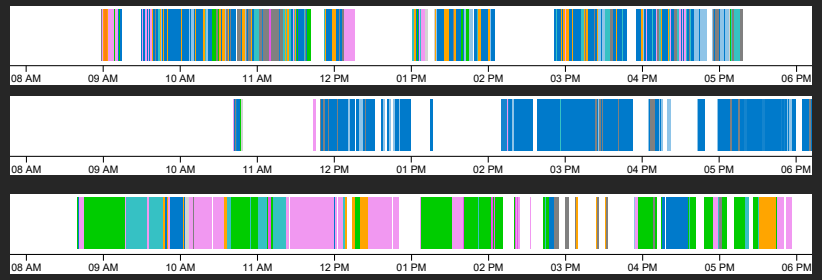
49% less interruptions
85% continued using it on a daily basis

BBC WORLD NEWS
THE NEW YORKER
New Scientist
THE TIMES
WSJ
cnBeta.COM 中文业界资讯站
20 minuten
NRJ
GeekWire
DIGITAL TRENDS
The Telegraph
NBC NEWS

Take aways & ideas

Productivity…

- Is affected by high work fragmentation
- Is individual, yet related to the team
- Follows habitual patterns



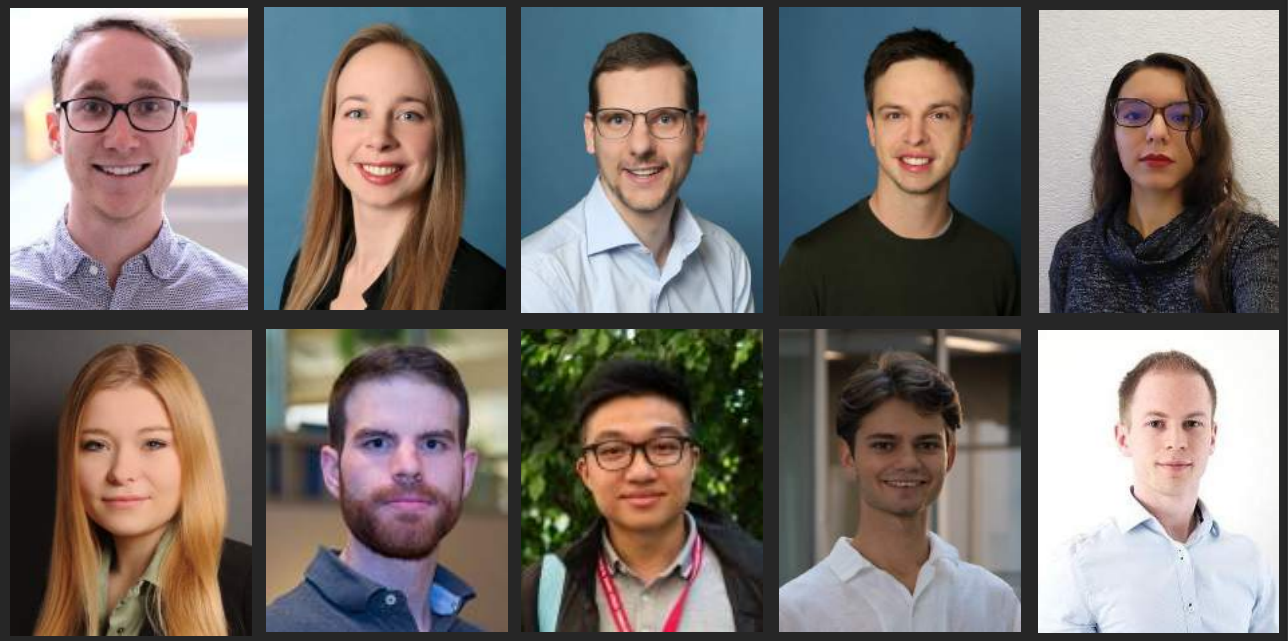
Boost it by…

- Tracking your own productivity curve & reflection
- Automatically sharing your interruptibility with your team (FlowLight)
- Creating and cultivating a team mindset
- …

HASEL

hasel.dev

University of Zurich[UZH]

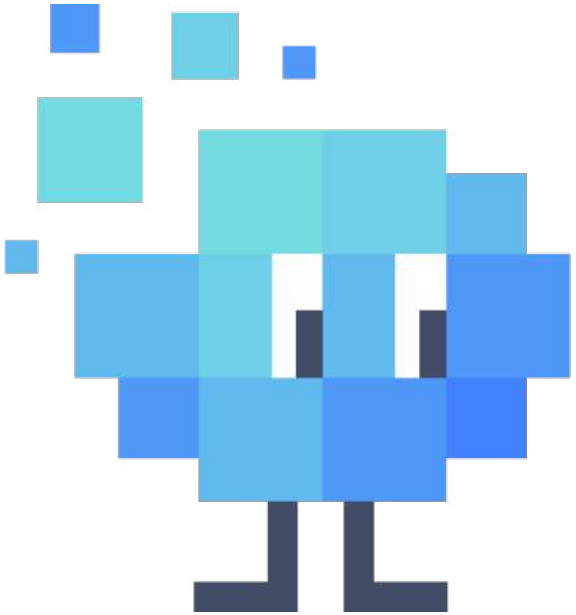


- Understanding human and social aspects of developers at work
- (Biometric) sensing of cognitive and emotional states
- Developing support to foster productivity and well-being
- Supporting information needs and reducing distractions

Further collaborators:
G. Murphy, T. Zimmermann, L. Howe, D. Shepherd, L. Jäger, S. Müller, K. Kevic, M. Züger, C. Satterfield, …

CREDIT SUISSE · Microsoft · interaxon · quickmobile · adesso · TRAIN WITH HEART · Schindler · logitech · ABB · IBM · VENTYX · Tasktop · BISON · iQmetrix

End of Excursion



HASEL

Human Aspects of Software Engineering Lab

Teaching & Course Assistants

Jonas Blum jonas.blum2@uzh.ch

Sijing Qin sijing.qin@uzh.ch

Nils Grob nils.grob@uzh.ch

Teaching & Course Assistants

Tarek Alakmeh tarek.alakmeh@uzh.ch

This course is about…

- Software engineering, software processes, steps in software development, **design and modularity**, testing and more
- By the end you should be able to
  - Demonstrate the knowledge to explain challenges of the steps involved in building a large, complex software system
  - Understand and justify benefits and disadvantages of various designs (high- and low-level) for constructing software systems
  - Communicate technical matters with programmers, managers, and clients effectively; and **convey the rationale behind your choices**

“Software is eating the world”

– Marc Andreesen

Tentative Course Schedule

Note: March 20th,

18:15 – 20:00 test exam

21.02	Introduction on Software Engineering
28.02	Process & Agile
06.03	Requirements Specification & User Stories
13.03	Design – Modularity & High-level Design
20.03	Design – Lower-level Design & Decomposing User Stories
27.03	Designing APIs & REST APIs
03.04	*Easter*
10.04	**Midterm [Room KOH-B-10]**
17.04	Modular Design & Design Principles (SOLID)
24.04	Design Patterns & Architectural Styles
01.05	Testing and Software Quality
08.05	Code Smells and Refactoring
15.05	Software Evolution
22.05	Cloud Deployment & Release
29.05	User Interface Design, Usability and Bias
19.06	**Final [Room HAH-E-03, and others]**

SE and SoPra

- SoPra is the application of the concepts taught in SE
- Not possible to completely sync it, but we try our best and are continuously improving it
  - Please do not expect it, some material will be covered here in more detail, only after it was part of SoPra
  - I will try to put relevant material up beforehand if possible

- Also keep in mind: not everyone takes both

SE and Software Construction

- Some repetition on purpose, and sometimes also with a different focus and detail

Prior Knowledge & More

Students should have completed software construction

You should be familiar with Object-Oriented Programming

This course will use Java for examples

Lecture sessions – somewhat *flipped* classroom

Each week before class (if specified)

- **Watch videos** (youtube) **/ read readings** (see OLAT)

  Videos/readings mostly from Reid Holmes (colleague at UBC)

  Video content is lecture material

Lecture sessions – somewhat *flipped* classroom

Each week before/at beginning of class (10am – 10:20am)

- Answer online OLAT quiz/survey (*individually*)
- You have **5mins** once you start (as long as you start before 10:15)
- These questions are for you to practice the material and focus on the prepared material

- If you have any questions about the material, state it in the quiz/survey as well

Lecture sessions – somewhat *flipped* classroom

Each week in class

- Recap
- I will try to integrate multiple activities (have your laptop and also a Java IDE available)
- Should be great practice for midterm and final

➢ I'll try to have more practice, less listening

➢ there will still be adjustments from last term, please bare with me

My Expectations

Be professional

questions in class, email, interacting with me/us

Attend lectures

talk to classmates if you are away

Participate

during activities, discussions, surveys

what you get out of this class will be directly proportional to how much you participate

Grading

30% Midterm BYOD (in person) (10.4, 10am – 12pm)

65% Final BYOD (in person) (19.6, 10am – 12pm)

midterm & final most likely multiple choice (Kprim questions)

5% Participation (weekly survey, at least for most weeks)

each quiz/survey has ~4 questions, one attempt to answer it

5 minutes time to answer

4 points per week, overall quizzes **>=75%** correct → 6.0 for this part

questions similar, grading slightly different to midterm/final

Previous classes

Quantitative answers

- What is the output of this code? “27”
- What is the time complexity? “O(n)”
- Which data structure should we use? “BST”

There is a right answer!

60

This class

Qualitative answers

- How good is this design?
- How good is this elicitation question?
- What problems could you encounter?
- How could you deal with problem *X*?

No right answer

But *great*, *ok* and *bad* answers

61

Resources

Website https://hasel.dev/teachings/fs24-se/

OLAT https://lms.uzh.ch/auth/RepositoryEntry/17509057911/

- Links to videos / readings
- Recordings
- Forum (do NOT post solutions and do NOT use it for personal issues that need to be discussed with me)
- Other material

In class resources will **NOT** be distributed

Reading on course / videos are essential for lectures

Class Activity

Analyzing Software Organizations

by Amy J. Ko

https://canvas.uw.edu/courses/1345618/pages/activity-analyzing-software-organizations

Analyzing software organizations

Every software organization has a role in the world; usually an economic one, but also likely a functional one. As software engineers, a critical skill is to be able to analyze a software organization and understand its role. This is useful if you're trying to decide whether to *join* an organization, but also if you're *starting* one.

This analysis skill involves doing the following

1. Analyze what software the organization makes.
2. Analyze what value that product offers.
3. Analyze how the organization is funded.
4. Analyze who the product is intended for.
5. Analyze the relationship between the organization's funding and the software's value.
6. Analyze what the software automates.
7. Identify other organizations that offer the same value.
8. Analyze the ethical values promoted by the organization and its software.
9. Synthesize all of the above into a story about the organization's role in society.

Analyzing Uber

1. Uber makes software to match drivers to passengers.
2. This allows passengers to get rides quickly and drivers to maximize earnings.
3. Uber takes a cut of the fare that passengers pay, and more if there's high demand or low supply.
4. Uber is for drivers and passengers.
5. The faster drivers and passengers are matched, the more rides happen, and the more Uber makes.
6. Uber automates the dispatcher role at taxi companies (the person you call for a ride)
7. Taxi companies and Lyft do the same thing as Uber.
8. Uber defines drivers as contractors rather than employees, which enables it to evade many workplace regulations on employers.
9. Uber's goal for society is to get people into rides faster. In the process, they will eliminate dispatcher jobs and the taxi companies that employ them. If they're successful, there will be no taxi companies other than companies like Uber, fewer jobs in the taxi industry, and faster access to rides.

With this analysis, you as a future software engineer can ask:

- Do you believe the future envisioned by the company is a good one?
- Do you want to automate what they help automate?

Activity: analyze for 15 mins (in groups)

Form a group of **two to four**. Select a software company that you'd all like to understand better. Conduct the analysis above, researching their websites, their funding, their products, and write answers to the nine prompts (+ which company you chose).

https://bit.ly/3T3gNnu

Next week

Software Processes

Things to do THIS WEEK

- ***Register*** (Modulbuchung) → you will be registered for OLAT (campus course) automatically
- ***Read*** the 'introduction to software engineering' (Reid) and the 'brief history of software engineering' (by Amy J. Ko)
- *Optional: listen to programming language introduction video*

- ***Listen*** to the Process module videos; additionally provided reading also captures most of the video material
  - Why Process?, Traditional Processes (Waterfall & Spiral), Agile Processes (XP and TDD), Scrum → 6 videos (1 to 6mins long)

- ***Fill out first quiz/survey on OLAT Wednesday (10-10:20)***