CSYE 7230
Software Engineering

Abdulkareem Alali

Syllabus and Plan

aalali.github.io/sef24-csye7230

Questions

1. Name?

2. Major?

3. Programming Languages?

4. Work outside of college?

5. Future best job?

6. Interesting thing about you?

Software Engineering

We know that computers and software **automate** many tasks

- Hardware quality is important?

- What about **Software Quality**?

"Software entities are more ==complex for their size== than perhaps any other ==human construct== because ==no two parts are alike== (at least above the statement level).

If they are, we make the two similar parts into a subroutine — open or closed. In this respect, software systems differ profoundly from computers, buildings, or automobiles, where repeated elements abound."

(No Silver Bullet, Brooks 1995, p 82) . (read the paper)
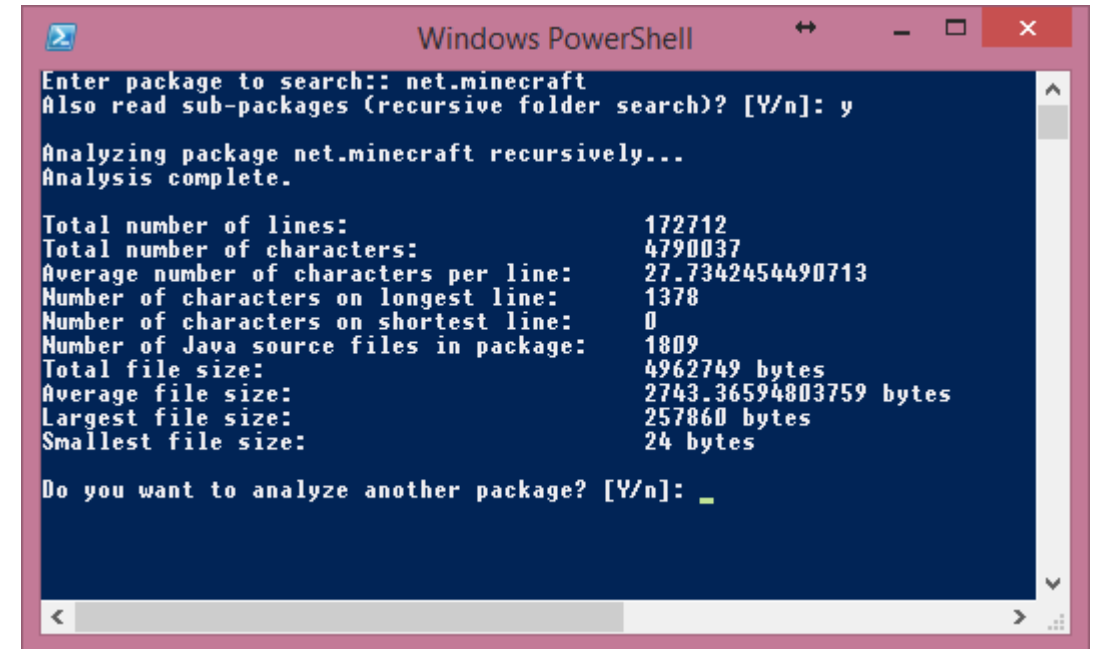
Large Software?

How many lines of code was used to make Minecraft?

By putting the 1.9.4 (2016) source through the java decompiler, recursively count the lines, decompiled jar yields 603,343 LOC .

Minecraft Forge Decompile 1.7.5 (2014) there is 172,712 lines, no comments .

Video game **Doom, Quake**?

Doom 3 has 601k, Quake III has 229k and Quake II has 136k .

```
Windows PowerShell

Enter package to search:: net.minecraft
Also read sub-packages (recursive folder search)? [Y/n]: y

Analyzing package net.minecraft recursively...
Analysis complete.

Total number of lines:                  172712
Total number of characters:             4790037
Average number of characters per line:  27.7342454490713
Number of characters on longest line:   1378
Number of characters on shortest line:  0
Number of Java source files in package: 1809
Total file size:                        4962749 bytes
Average file size:                      2743.36594803759 bytes
Largest file size:                      257860 bytes
Smallest file size:                     24 bytes

Do you want to analyze another package? [Y/n]: _
```

Large Software?

How many lines of code are in Android, Windows, Firefox, Chrome?

- Windows 3.1 ~ 3 million lines of code

- Google Chrome has about ~ 5 million LOC

- Firefox ~ 10 MLOC

- Android ~ 12 MLOC

- Windows 7 ~ 40 MLOC, a little less than Windows XP and ~10 MLOC less than Windows Vista .

- Windows 10 ~ 50–60 MLOC .

- Windows 11 ~ 60–100 MLOC .

Large Software?

IEEE Spectrum article by Robert Charette entitled: "[This Car Runs on Code](#)," 2009:

- First car to incorporate embedded software was the 1977 General Motors Oldsmobile Toronado which had an electronic control unit (ECU) that managed electronic spark timing [.](#)

- By 1981, GM had deployed about 50,000 lines of engine control software code. Others soon followed the trend.

- Volt has ~ 10 million lines of code

- New Mercedes-Benz E-Class has more LOC than an Airbus A380 [.](#)

- A modern high-end car features around 100 million lines of code, and this number is planned to grow to 200-300 millions in the near future [.](#)

Codebases

Millions of lines of code

| hundred thousand | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |

simple iPhone game app

Unix v 1.0
1971

OPERATING SYSTEM

Win32/Simile virus

average iPhone app

APP

Pacemaker

Photoshop v. 1.0
1990

Camino
web browser

BROWSER

Quake 3 engine
3D Video game system

GAME

Space Shuttle
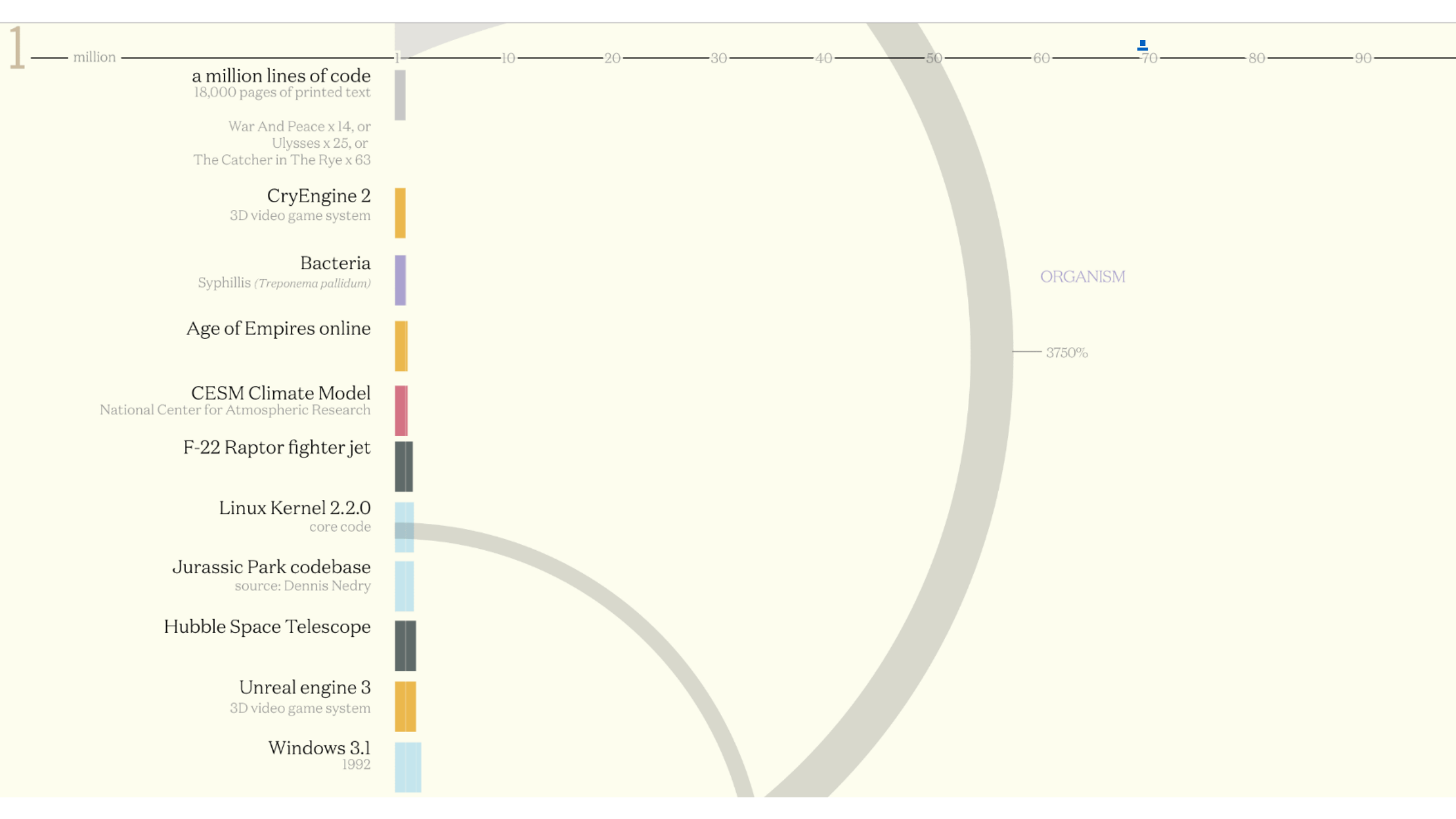
MACHINE

a million lines of code

1 million

| 1 | 10 | 20 | 30 | 40 | 50 | 60 | 70 | 80 | 90 |

1 million — 1 — 10 — 20 — 30 — 40 — 50 — 60 — 70 — 80 — 90

a million lines of code
18,000 pages of printed text

War And Peace x 14, or
Ulysses x 25, or
The Catcher in The Rye x 63

CryEngine 2
3D video game system

Bacteria
Syphillis *(Treponema pallidum)*

Age of Empires online

CESM Climate Model
National Center for Atmospheric Research

F-22 Raptor fighter jet

Linux Kernel 2.2.0
core code

Jurassic Park codebase
source: Dennis Nedry

Hubble Space Telescope

Unreal engine 3
3D video game system

Windows 3.1
1992

ORGANISM

3750%

Windows NT 3.1
1993

5

HD DVD Player on XBox
(just the player)

needed to repair HealthCare.gov
apparently

WEB

Mars Curiosity Rover
Martian ground vehicle probe

Linux kernel 2.6.0
2003

166%

Google Chrome
latest

upper
estimate

World of WarCraft
server only

Boeing 787
avionics & online support systems only

Windows NT 3.5
1993

Firefox
latest version

10

Chevy Volt
electric car

Intuit Quickbooks
accounting software

288%

Windows NT 4.0
1996

Android
mobile device operating system

Android
mobile device operating system

Mozilla Core
core code at heart of all Mozilla's software

MySQL
database language

Boeing 787
total flight software

Linux 3.1
recent version

Apache Open Office
open-source office software

F-35 Fighter jet
2013

Microsoft Office 2001

Windows 2000

Microsoft Office for Mac
2006

Symbian
mobile operating system

Windows 7
2009

Windows XP
2001

25

252%

180%

138%

1   10   20   30   40   50   60   70   80   90

50

	Windows XP	2001
	Microsoft Office 2013	
	Large Hadron Collider	total code
	Windows Vista	2007
	Microsoft Visual Studio 2012	
	Facebook	(including backend code)
	US Army Future Combat System	fast battlefield network system (aborted)
	Debian 5.0 codebase	free, open-source operating system
	Mac OS X "Tiger"	v 10.4

100

| | Car software | average modern high-end car |
| | Mouse* | Total DNA basepairs in genome |

2 billion

125%

Scale markers: 1 · 10 · 20 · 30 · 40 · 50 · 60 · 70 · 80 · 90

1 10 20 30 40 50 60 70 80 90

Google
all internet services
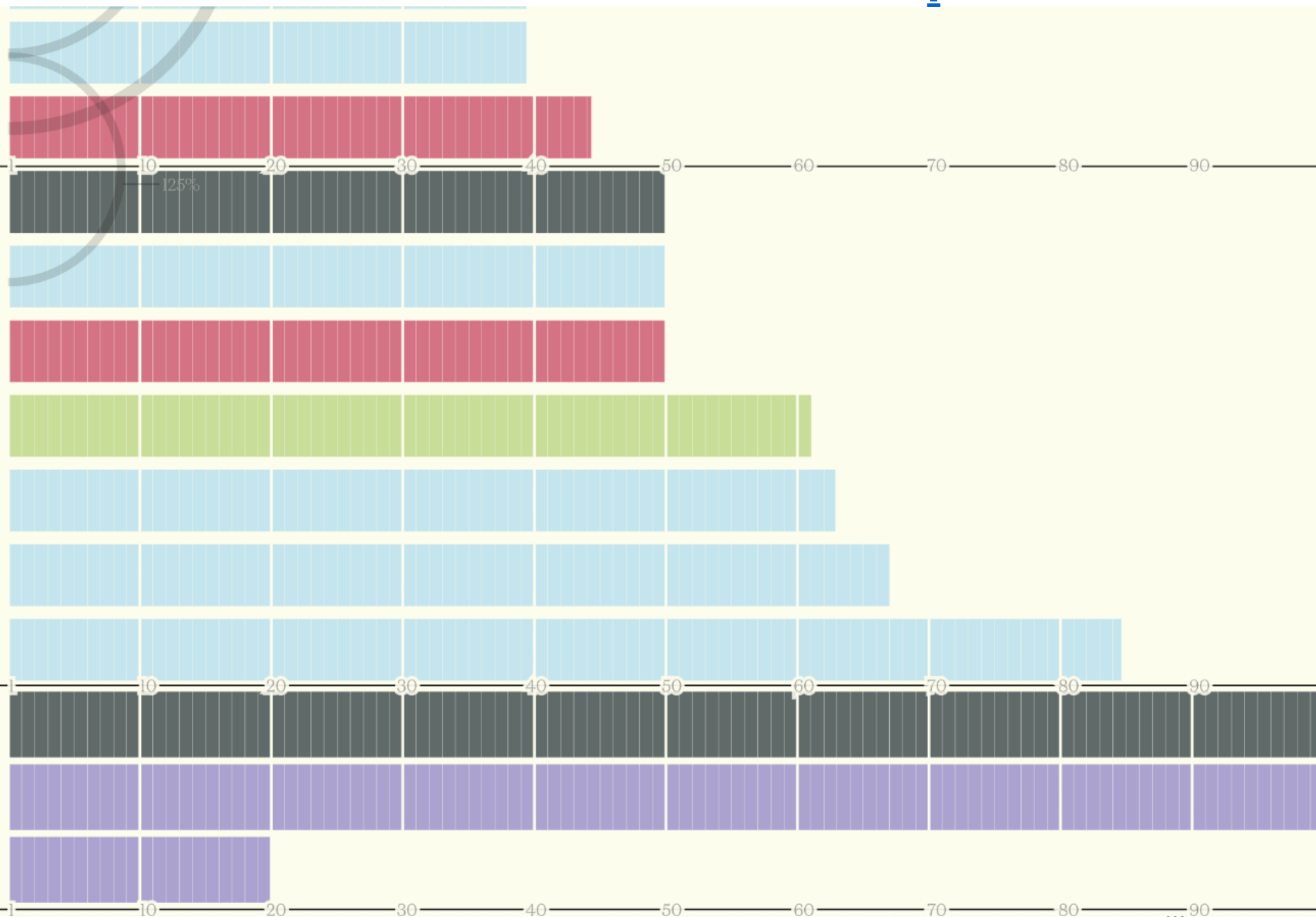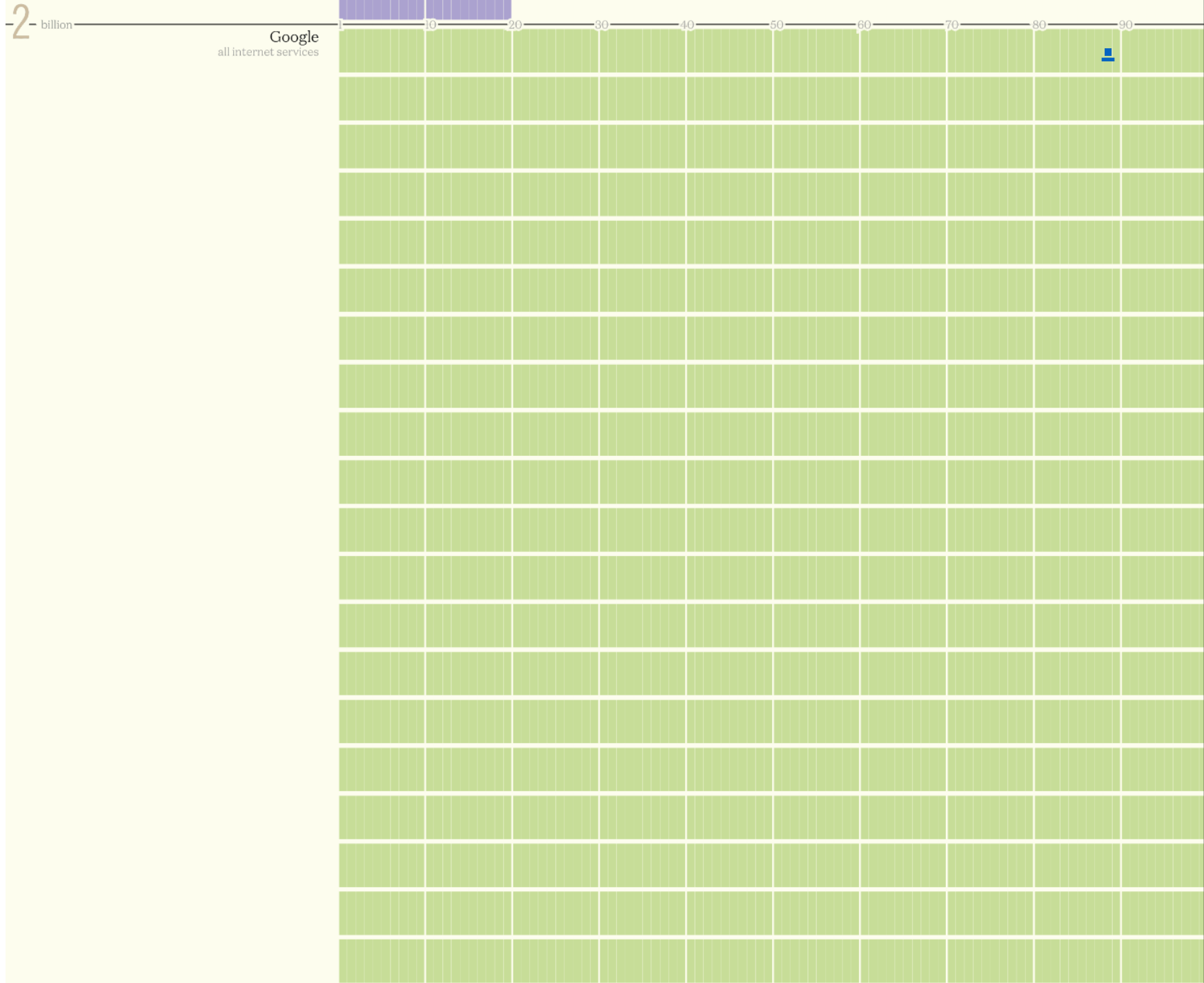
*Human Genome = 3,300 billion "lines" of code

concept & design: David McCandless
informationisbeautiful.net
research: Pearl Doughty-White, Miriam Quick

this graphic is a part of
knowledge is beautiful bit.ly/KIB_Books
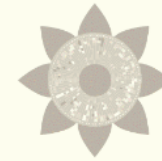
sources NASA, Quora, Ohloh, Wired & press reports
note some guess work, rumours & estimates
data bit.ly/KIB_linescode

v 0.9// Sep 24th 2015

Information is Beautiful
Book by David McCandless

The mouse genome is contained in 20 chromosome pairs and the current results suggest that it is about 120 billion base pairs in size (DNA nucleobases A, G, C, and T)

The human genome is 3,300 billion base pairs spread out over 23 pairs of chromosomes

How could we compare LOC to base pairs?

What's Involved in Large-Scale Software Development?

Assume you and a few friends are going to undertake a substantial software project.

What are the things that need to be done?

How would you start?

What other considerations would you have?

Large Projects Need Planning

On small programming projects there is often a single programmer who knows, or thinks he/she knows, what needs to be done

The focus is on implementation

Large-scale projects require more planning and this is what software engineering is about!

Unfortunately there is no known approach that, if followed, will guarantee success

General Problems - The Nature of Software

50% of all software projects fail

- Never delivered/completed
- Do not meet requirements or user needs
- Excessive failures (bugs)
- Excessively over budget or late

Quality and reliability of many software systems can not be formally assessed

Software Production Problems .

Complexity

- Interacting parts in a software system. Functions or objects, and invoked as needed rather than being replicated

- Software parts have several different kinds of interactions, including serial and concurrent invocations, state transitions, data couplings, and interfaces to databases and external systems

Software Production Problems [.](#)

Conformity

- Lack of conformity can cause problems when an existing software component cannot be reused as planned because it does not conform to the needs of the product under development

- Lack of conformity might not be discovered until late in a project

- This requires an unplanned allocation of resources (usually) and can delay project completion

Software Production Problems .

Changeability

- Software can "easily" be changed, but a bridge is almost impossible to move

- However, this does not mean that software is easy to change, side effects!

Software Production Problems [.](#)

Invisibility

Software is very hard to visualize

Effects of executing software on a digital computer are observable, software itself cannot be seen, tasted, smelled, touched, or heard. My code isn't a living tree? Maybe a virtual tree?

Due to no physical presence, software engineers must use different representations at different levels of abstraction in an attempt to visualize. A diamond as a service?

Software Production Problems [.](#)

Brook's "No Silver Bullet" [IEEE Computer 9(4), 1987]
  - Software is very difficult to develop, and most likely will not get easier
  - Reuse is one solution suggested

Brooks insists that there is no one silver bullet -- "there is no single development, in either technology or management technique, which by itself promises even one order of magnitude [tenfold] improvement within a decade in productivity, in reliability, in simplicity."

Software Crisis **.** **.**

"Software Crisis" NATO Software Engineering Conference in **1968** at Garmisch, Germany

"The major cause of the software crisis is that the machines have become several orders of magnitude more powerful!

To put it quite bluntly: 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."

—Edsger Dijkstra, The Humble Programmer (EWD340), Communications of the ACM, 1972 ACM Turing Award Lecture

History's Worst Software Bugs [.](#)

1993 – Intel Pentium floating point divide

- Pentium chip to make mistakes when dividing floating-point numbers that occur within a specific range

- For example, dividing 4195835.0/3145727.0 yields 1.33374 instead of 1.33382, an error of 0.006 percent

- Although the bug affects few users, it becomes a public relations nightmare

- With an estimated 3 million to 5 million defective chips in circulation, at first Intel only offers to replace Pentium chips for consumers who can prove that they need high accuracy;

- eventually the company relents and agrees to replace the chips for anyone who complains. The bug ultimately costs Intel **$475 million**

History's Worst Software Bugs .

2024 – CrowdStrike and Windows

- ~25% of Fortune 500 companies experienced disruptions due to the CrowdStrike outage

- Airlines, Healthcare, and Banking most impacted

- Estimated Financial Loss due to the CrowdStrike outage: **$5.4 billion** for the Fortune 500, excluding Microsoft

Software Development .

Better quality of the software development process has given rise to the discipline of

Software Engineering,

which aims to apply the **systematic** approach exemplified in the **engineering** paradigm to the **process of software development**

Software Engineering

(1) The **application** of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software; that is the **application of engineering to software**. (2) The **study** of approaches as in (1). (IEEE/Pressman)

Software engineering is the application of engineering to the design, development, implementation, testing and maintenance of software in a systematic method .

We need discipline but we also need Adaptability and Agility

What is Software Engineering? •
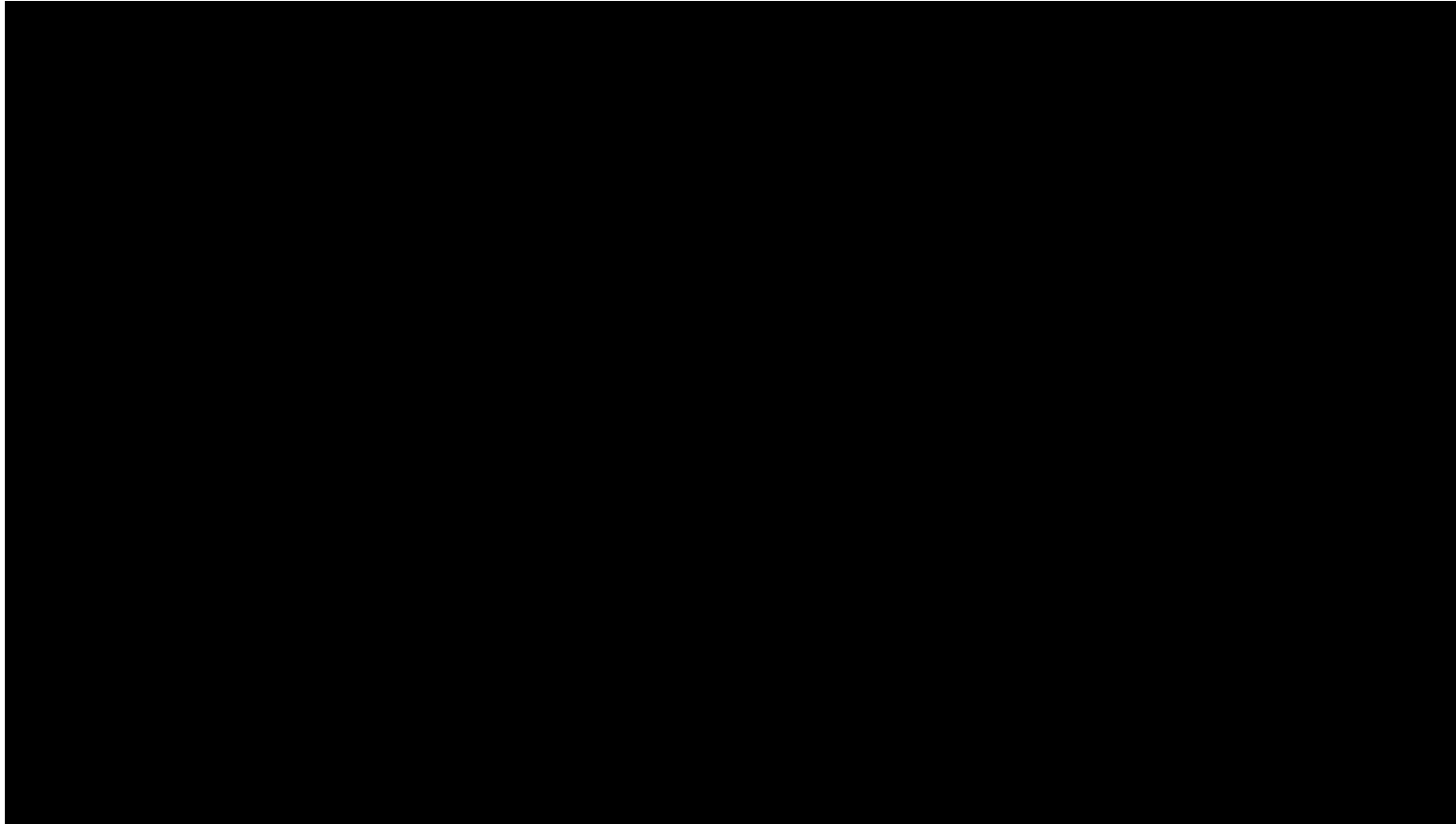
Software Engineering is a Layered Technology (Pressman)

Software Development Overview

Software Engineering is a **layered technology**. Any engineering approach must rest on an organizational commitment to **Quality**.

The bedrock that supports software engineering is **a Quality focus**. (Pressman)

tools

methods

process model

a "quality" focus

Software Development Overview

Quality Focus: Focus on certain aspects of quality

Process: A sequence of activities, people, and systems involved in achieving some desired result

- Glue that holds technology layers together and enables timely development of software
- Management control of software projects
- Technical **methods** are applied to construct a product:
    - Milestones are established,
    - Quality is ensured,
    - And change is properly managed

Software Development Overview

Methods: software engineering methods provide the technical **how to** for building software.

**Requirement gathering,
analysis,
design,
coding,
testing, and
support/maintenance (~80%).**

Tools: Tools provide support for the methods and process
- Automated or semi-automated support for the process and the methods

Process

Software development life-cycle is the phases a software product goes through between when it is conceived and when it is no longer available for use

Methods

Software Development Methodology (SDM)

Requirements

- Requirements are determined

- Use cases are done

- A high-level functional specification is created

Analysis

- Conceptual classes are identified

- Estimations for time, risk analysis

Methods

Software Development Methodology (SDM)

Design

- Abstractions of the system is created
- System Architecture is defined
- Objects will be determined (some of this in analysis also)
- Interfaces are defined
- Data structures may be chosen
- Algorithms may be chosen

Methods

Implementation

- If design is done well, implementation is straight-forward

Testing

- Testing should be continuous and on-going

Release

- Software is released to customers

Methods

Maintenance (80-90% cost) .

Modifies the software for the following purposes:

- Adaptive - modify to accommodate changes in the external environment
- Corrective - fix errors
- Perfective - add additional functionality
- Preventive - make changes to facilitate easier correction, adaptation, and enhancement

Process and **Methods**

Systems Development Life Cycle (SDLC) and Software Development Methodology (SDM)

The divisions are not necessarily clear-cut and they affect each other

There is usually **feedback** from latter stages to earlier stages

Some things, like **documentation**, are done in most or all stages

The Process

Software Development Process Models or SDLC

There are a number of process models

For example the **Waterfall Model** or **linear sequential** model process is essentially the steps of the software development life-cycle. This approach does not work, maybe, or does it? . . .

Usual development processes run **iteratively** and **incrementally** through the software development life-cycle phases rather than linearly

Software Development Process Models .

Waterfall

Prototyping

Iterative and Incremental development
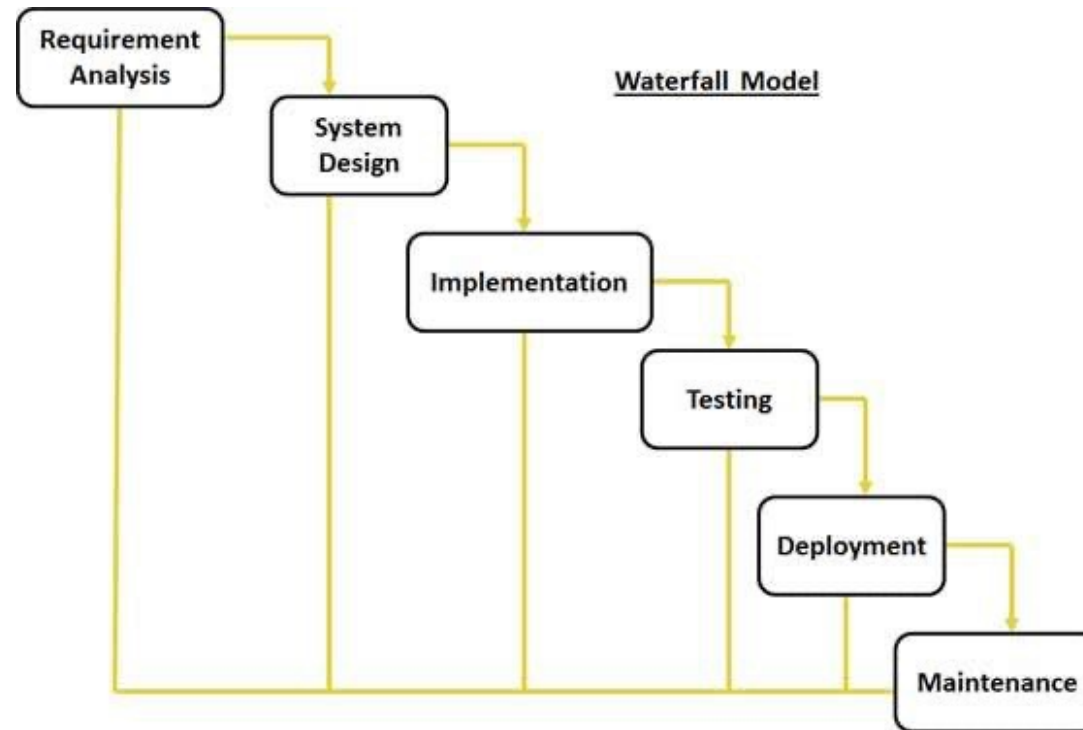
Spiral development

Rapid application development

Extreme programming

Agile methodology

...
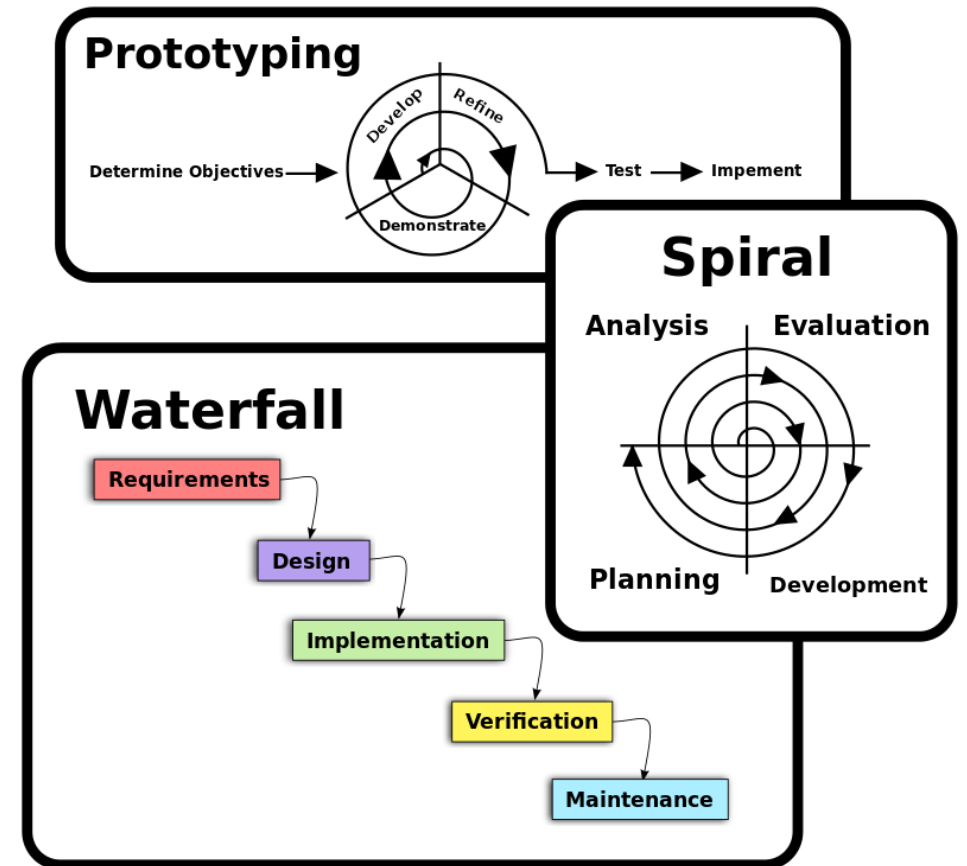
Waterfall Model Design .

Waterfall Model Design .

Requirement Gathering and Analysis

ALL possible requirements of the system to be developed are captured in this phase and documented in a requirement specification doc

System Design

The requirement specifications from first phase are studied in this phase and system design is prepared

Waterfall Model Design **.**

Implementation

With inputs from system design, the system is first developed in small programs called units, which are integrated in the next phase. Each unit is developed and tested

Integration and Testing

All the units developed in the implementation phase are integrated into a system after testing of each unit. Post integration the entire system is tested for any faults and failures

Waterfall Model Design .

Deployment of System

Once the functional and non functional testing is done, the product is deployed in the customer environment or released into the market

Maintenance

There are some issues which come up in the client environment. To fix those issues patches are released. Also to enhance the product some better versions are released

Waterfall Model Application

- Requirements are very well documented, clear and fixed

- Product definition is stable

- No ambiguous requirements

- Resources with required expertise are available

- The project is, usually, short

Waterfall Model Pros.

Simple and easy to understand and use, easy to manage

Phases are processed and completed one at a time

Works well for smaller projects where requirements are very well understood

Clearly defined stages, Easy to arrange tasks

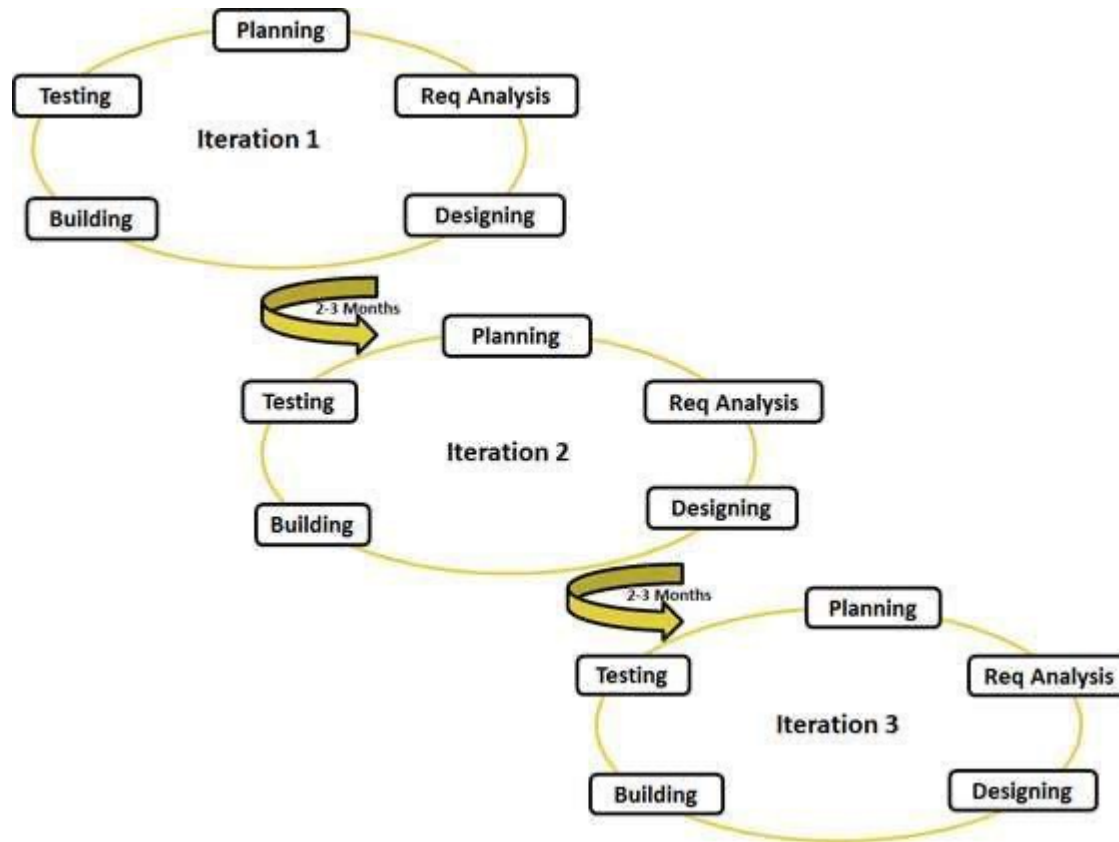Process and results are well documented

Waterfall Model Cons.

No working software is produced until late during the life cycle

Not suitable for the projects where requirements are at a moderate to high risk of changing

It is difficult to measure progress within stages

Integration is done as a big-bang at the very end

Agile Model .

Agile Model .

Individuals and Interactions

- self-organization and motivation are important

Working software

- Working software is considered the best means of communication with the customer to understand their requirement, instead of just depending on documentation

Agile Model .

Customer Collaboration

- As the requirements cannot be gathered completely in the beginning of the project

- Continuous customer interaction is very important to get proper product requirements

Responding to Change

- Agile development is focused on quick responses to change

Popular Agile Model Methods .

Rational Unified Process (1994),

Scrum (1995),

Crystal Clear,

Extreme Programming (1996),

Adaptive Software Development,

Feature Driven Development, and

Dynamic Systems Development Method (DSDM) (1995).

Agile Model Pros .

- Is a very realistic approach to software development
- Promotes teamwork and flexibility to developers
- Functionality can be developed rapidly and demonstrated
- Suitable for fixed or changing requirements and delivers early partial working solutions
- Enables concurrent development
- Delivery within an overall planned context

Agile Model Cons .

- Agile leader and management practice is a must
- Strict delivery management for functionality to be delivered, and adjustments to meet the deadlines
- Depends heavily on customer interaction, so if customer is not clear, team can be driven in the wrong direction
- There is very high individual dependency and transfer of technology to new team members may be quite challenging due to lack of documentation

Agile Vs Traditional SDLC
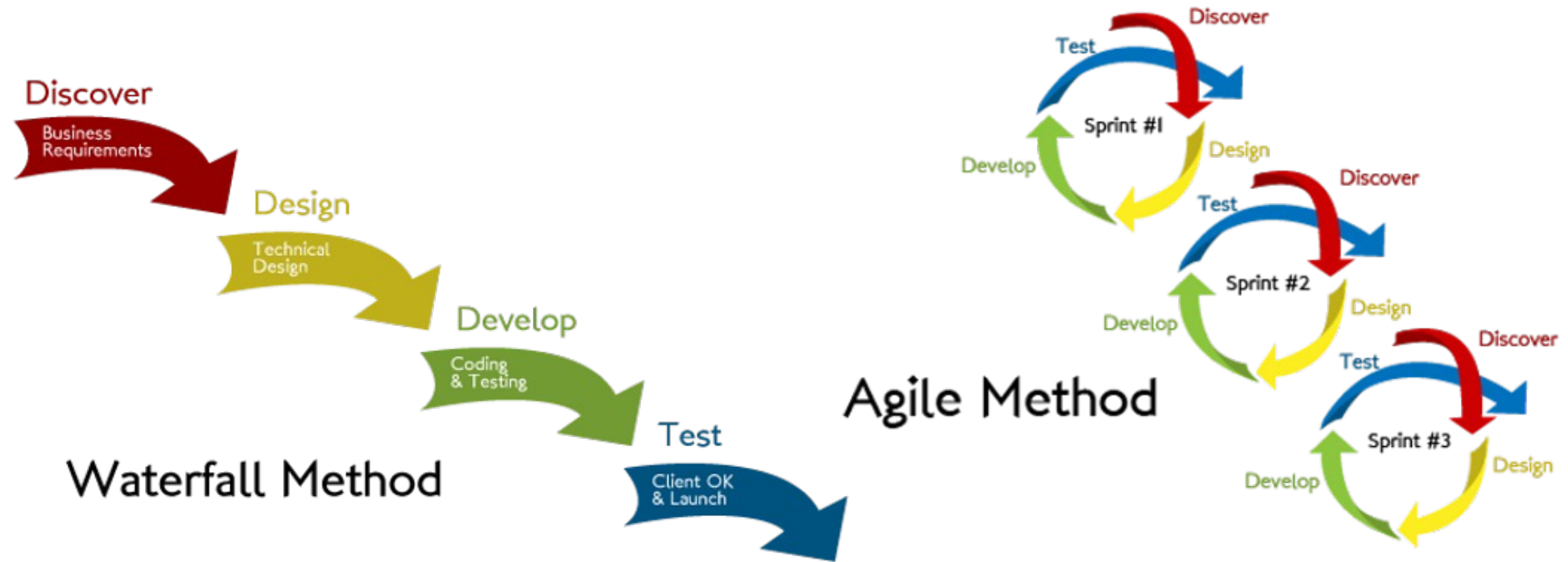
Agile Vs Traditional SDLC .
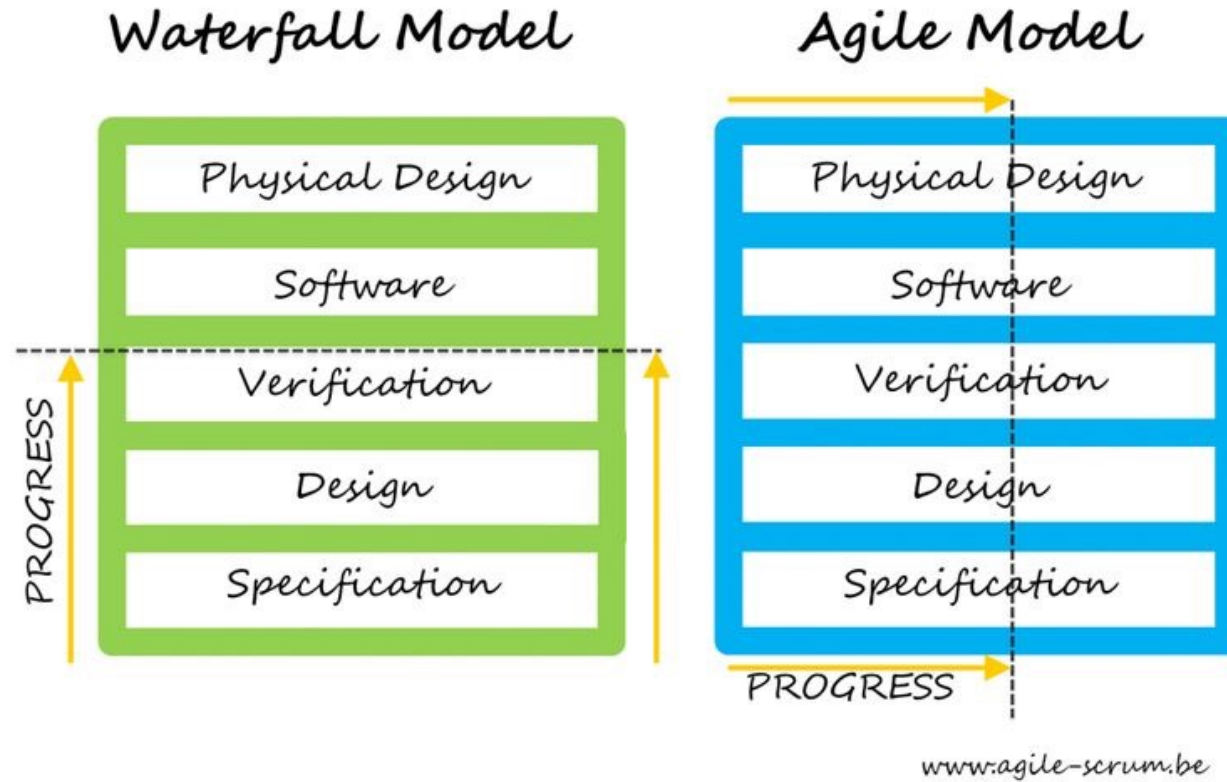
Agile Vs Traditional SDLC .

- Agile is **Adaptive**, waterfall model is a **Predictive** approach

- Predictive teams waterfall like models have a **complete forecast** of the exact tasks and features to be delivered

- Agile has no detailed planning and clarity on future tasks only in respect of what features need to be developed (%)

Agile vs. Traditional SDLC .

Agile team adapts to the changing product requirements dynamically

Tested very frequently, through the iterations, minimizing the risk of any major failures in future

Customer interaction is the backbone of Agile methodology, and open communication with minimum documentation

The agile teams often located in the same geographical location

Summary

Development of large-scale software is a complicated task

There are many aspects to the development of all substantial software systems

The software development life-cycle states the things that need to be done in a process but they can't just be done as a sequence, 1, 2, 3, … .