MASSEY UNIVERSITY
TE KUNENGA KI PŪREHUROA
UNIVERSITY OF NEW ZEALAND

Software Design and Construction 159.251 Introduction The Problem With Software

Amjed Tahir
a.tahir@massey.ac.nz

Adopted from material of Jens Dietrich

Readings

1. Brian Foote and Joseph Yoder: Big Ball of Mud.
   http://www.laputan.org/mud/mud.html#BigBallOfMud
1. Feature Creep.
   http://c2.com/cgi/wiki?CreepingFeaturitis
1. Martin Fowler: Technical Debt.
   http://martinfowler.com/bliki/TechnicalDebt.html
1. Have a quick look at Conway's game of life
   http://en.wikipedia.org/wiki/Conway's_Game_of_Life.

Common Problems with Software:

Software is everywhere (car, hospital, gym etc..)!

Software is too expensive, hard to manage, not easy to understand and its quality is often poor!

This may be related to the ever increasing size and complexity of software.

Let's explore this further ..

Spectacular Failures

Novopay botch-ups cost $45m to fix (2015)

Novopay is a web-based payroll system for state and state integrated schools in New Zealand, processing the pay of 110,000 teaching and support staff at 2,457 schools.

https://en.wikipedia.org/wiki/Novopay

The payroll botch-up, Novopay, that left thousands of teachers out of pocket has cost taxpayers an additional $45 million to fix.
..
(Ministry of Education secretary) Peter Hughes conceded Novopay had "cost us dearly".

http://www.stuff.co.nz/national/education/66349800/Novopay-botch-ups-cost-45m-to-fix

Spectacular Failures (ctd)

ARIANE 5 Flight 501 Failure (1996, estimated cost: USD 500,000,000)

On 4 June 1996, the maiden flight of the Ariane 5 launcher ended in a failure. Only about 40 seconds after initiation of the flight sequence, at an altitude of about 3700 m, the launcher veered off its flight path, broke up and exploded.

...

These nozzle deflections were commanded by the On-Board Computer (OBC) software on the basis of data transmitted by the active Inertial Reference System (SRI 2). **Part of these data at that time did not contain proper flight data, but showed a diagnostic bit pattern of the computer of the SRI 2, which was interpreted as flight data.**

...

http://www.di.unito.it/~damiani/ariane5rep.html
http://www.around.com/ariane.html

Spectacular Failures (ctd)

Computer problems hit three nuclear plants in Japan (2000)

Tokyo IDG Only a handful of computer problems have been reported in Japan in the new year to date; however, at least three hit systems associated with nuclear power plants, according to the government and power generating companies.

The potentially most serious problem occurred not at midnight but at 858 a.m. local time on Jan. 1 at the **Fukushima Number 2** nuclear power plant of Tokyo Electric Power Co. TEPCO. **The system that shows the position of the control rods in the reactor core failed, leaving operators unable to gauge the rods positions using the system ..**

http://articles.cnn.com/2000-01-03/tech/japan.nukes.y2k.idg_1_nuclear-power-plant-tokyo-electric-power-tepco?_s=PM:TECH

Spectacular Failures (ctd)

Boeing 737 max- when software and hardware interactions create a disaster! (2019)

A Lion Air and an Ethiopian Airlines (brand new!) 737 Max airplanes crashed shortly after takeoff, causing the death of 347 people.

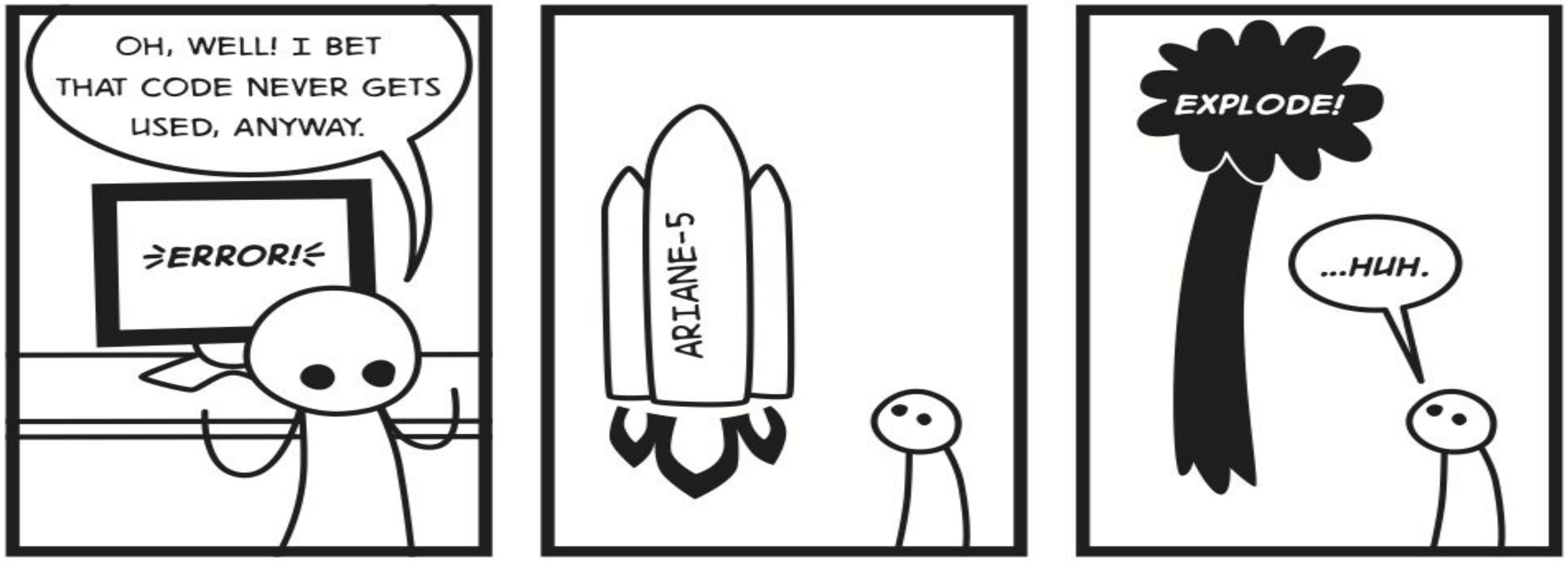
Boeing introduced the MCAS, which automatically nudges the nose down if onboard sensors detect that the plane risks stalling.**The software built in software was designed to work automatically and only in extreme situations**. Boeing decided pilots didn't need any new training to understand MCAS. In fact, they didn't even mention the system in flight manuals.

After the crashes, airlines grounded all their 737 max fleet (one of the most successful airplanes ever made!), and many cancelled future orders. **Boeing profit profit dropped 21 percent in the first three months of 2019** after software issues grounded its entire fleet of 737 Max.

"The flight management computer is a computer. What that means is that it's not full of aluminum bits, cables, fuel lines, or all the other accoutrements of aviation. It's full of lines of code. And that's where things get dangerous."

https://spectrum.ieee.org/aerospace/aviation/how-the-boeing-737-max-disaster-looks-to-a-software-developer

Ariane 5 - the most expensive bug in history!



A bug caused a data conversion from a 64-bit floating point number to a 16-bit signed integer value to overflow and cause a hardware exception

The Latest “Software” Disasters in Aviation!

Airbus issues software bug alert after fatal plane crash

US aviation authority: Boeing 787 bug could cause 'loss of control'

Pearls of Wisdom from Management

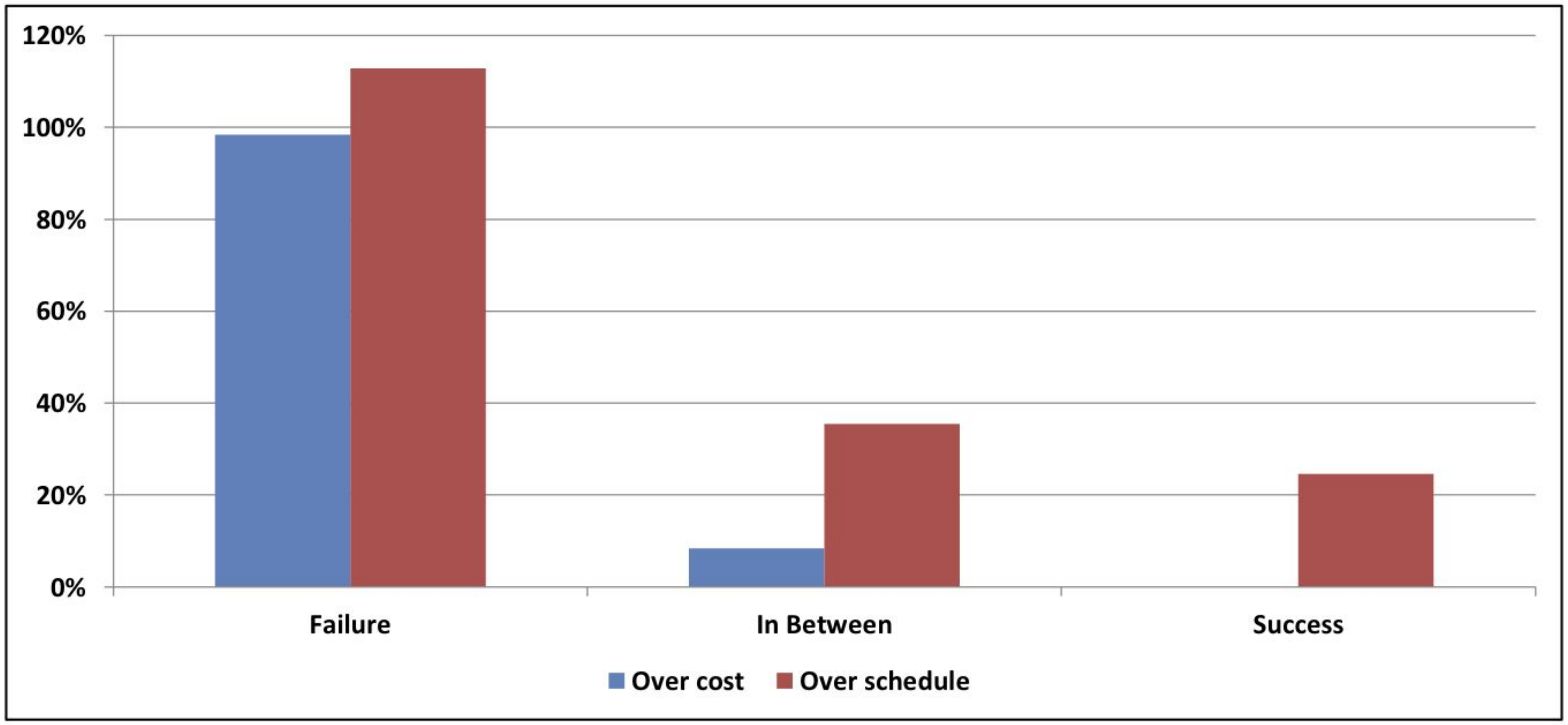
Frank Lanza (executive vice president of Lockheed Martin):

There is no life today without software. The world would probably just collapse. Fortunately, really important software has a reliability of 99.9999999 percent.
At least, until it doesn't.

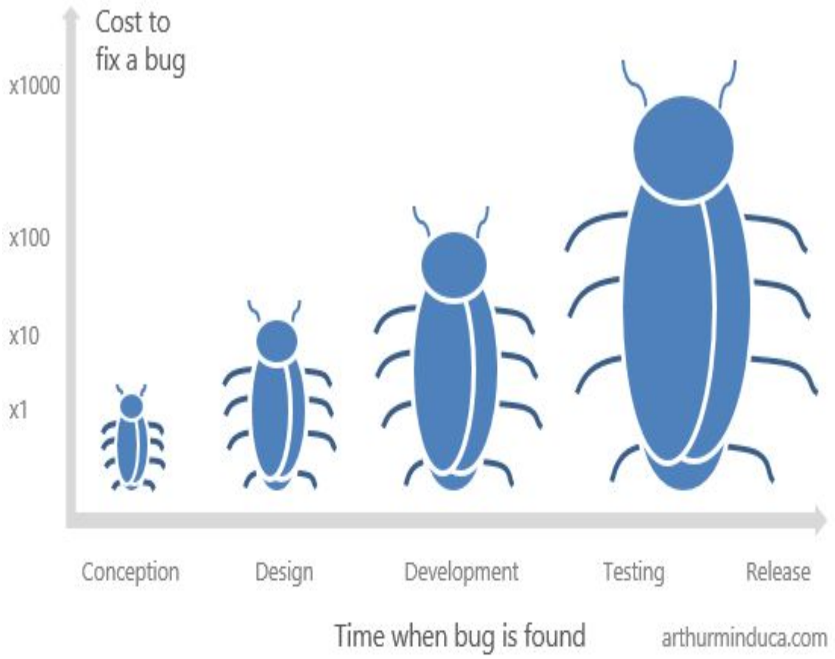
Cost

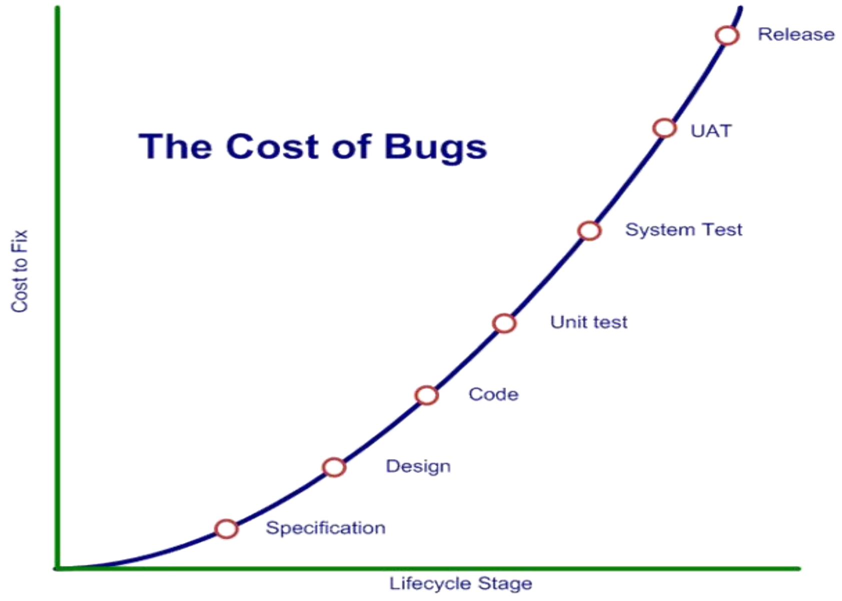
A recent study by Clarus (28 NZ organisations) suggests that **54%** of projects in the study **were failures** and **failed projects cost twice as much & take 110% longer than planned**

Cost and Schedule Overruns

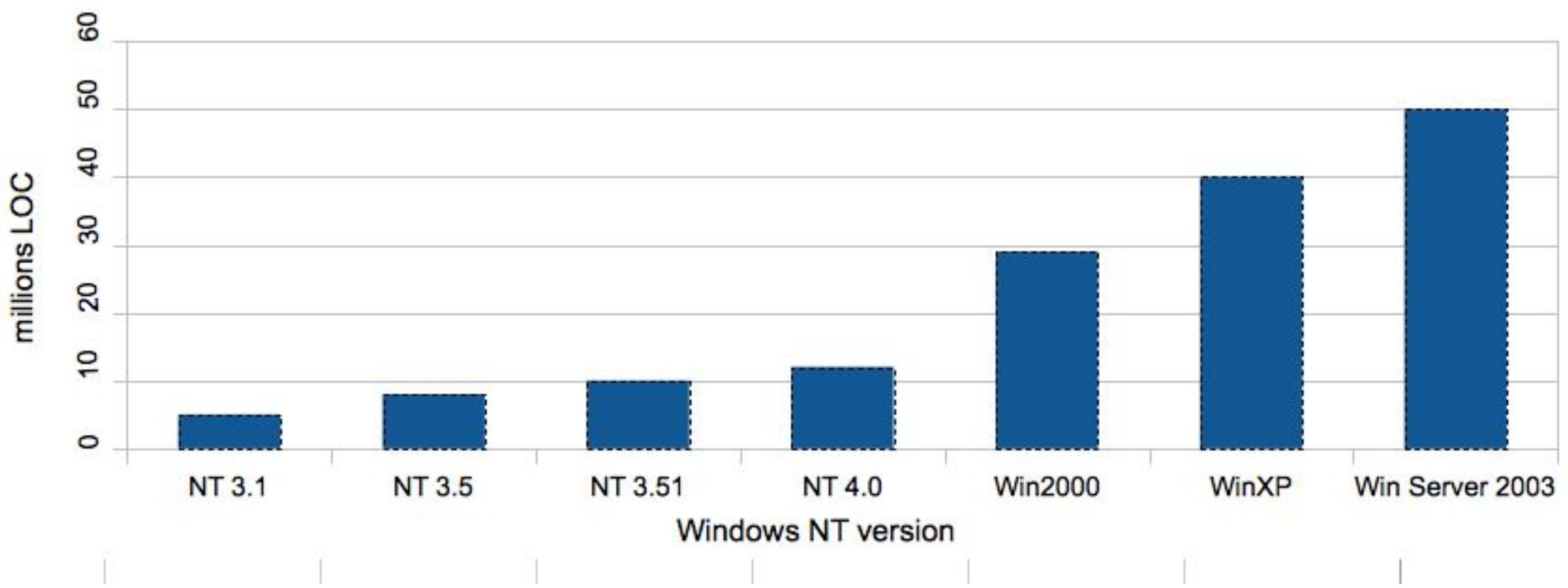


- average overruns for failed, in-between and successful projects

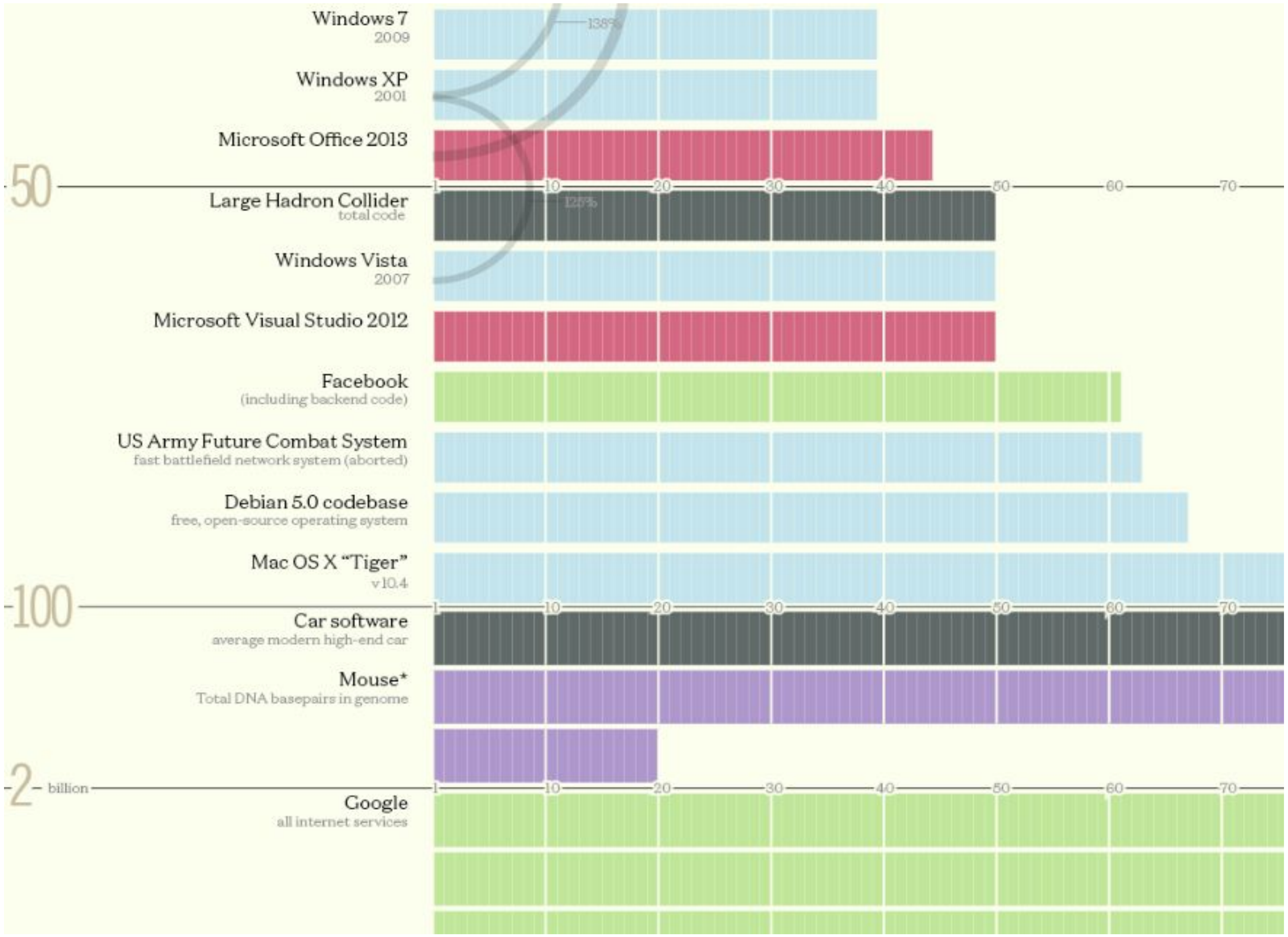

Cost to fix a bug
x1000
x100
x10
x1
Conception
Design
Development
Testing
Release
Time when bug is found
arthurminduca.com


The Cost of Bugs
Cost to Fix
Lifecycle Stage
Specification
Design
Code
Unit test
System Test
UAT
Release

The ever-increasing Size of Software



- the size of Windows NT versions from 1993-2003
- LOC: lines of code
- source: V. Maraia: The Build Master: Microsoft's Software Configuration Management Best Practices. Addison-Wesley 2005.



http://www.visualcapitalist.com/millions-lines-of-code/

What about Complexity ?

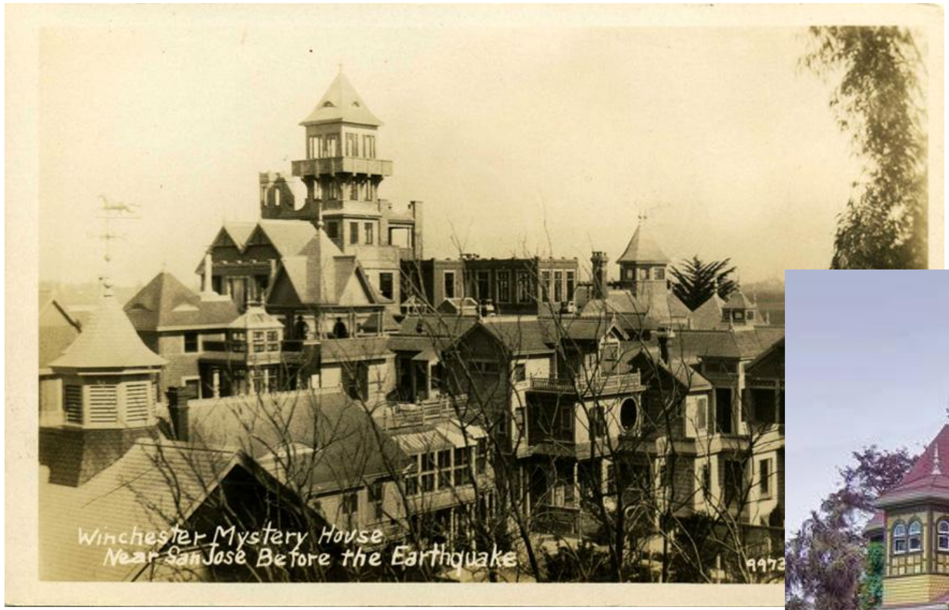
Complexity: **the state or quality of being intricate or complicated**.
source: New Oxford American Dictionary

The Winchester Mystery House

- Victorian mansion in California
- built 1884 - 1922
- cost apprx 5.5 million USD
- seven stories high, now 4 stories (after 1906 quake)
- there are roughly 160 rooms, including 40 bedrooms, 2 ballrooms (one completed and one unfinished) as well as 47 fireplaces, 10,000 window panes, 17 chimneys (with evidence of two others), two basements and three elevators.
- **a metaphor for complex, evolved design and quality problems resulting from this**

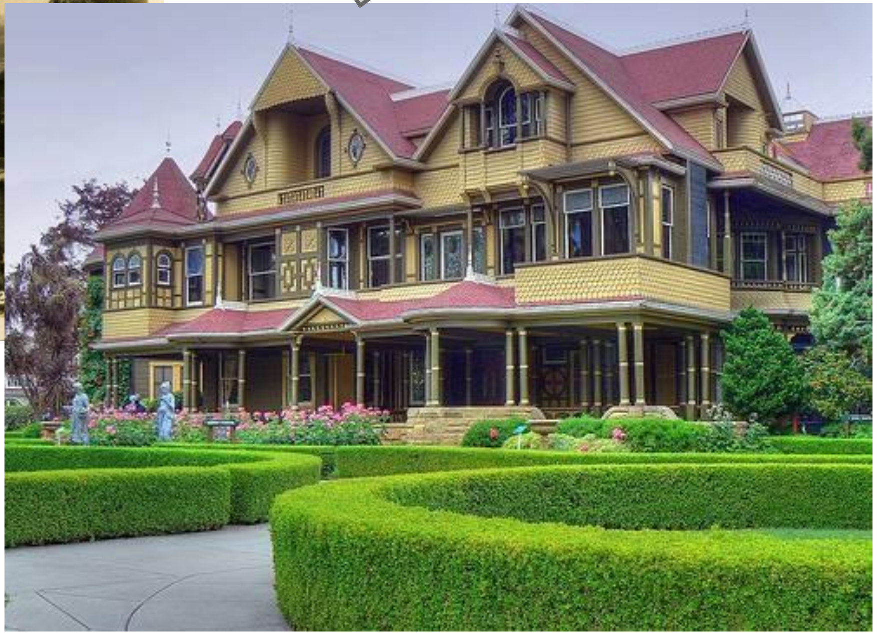
source: http://en.wikipedia.org/wiki/Winchester_Mystery_House

Winchester Mystery House



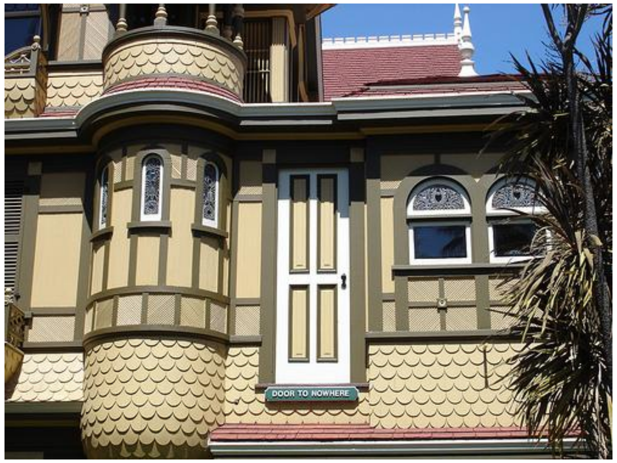
before the 1906 quake

after the 1906 quake



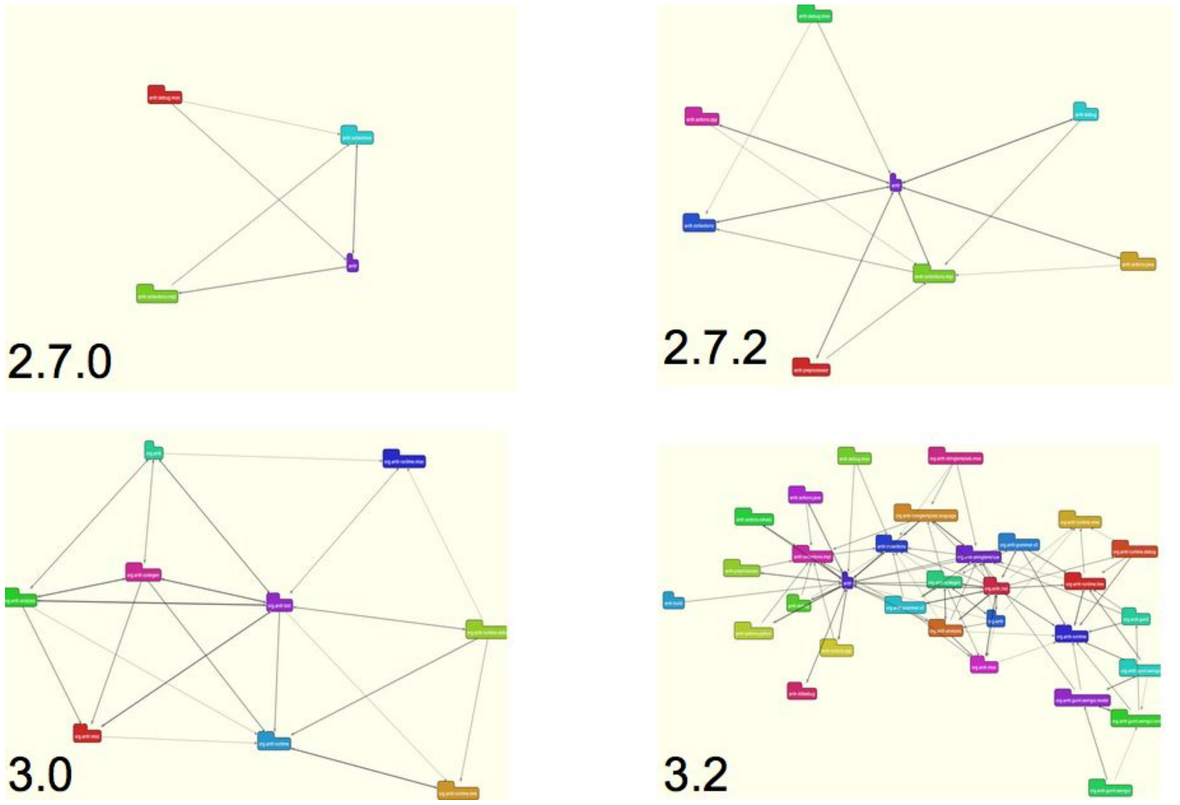
Winchester Mystery House Oddities

doors to nowhere



Fireplace that is not functional

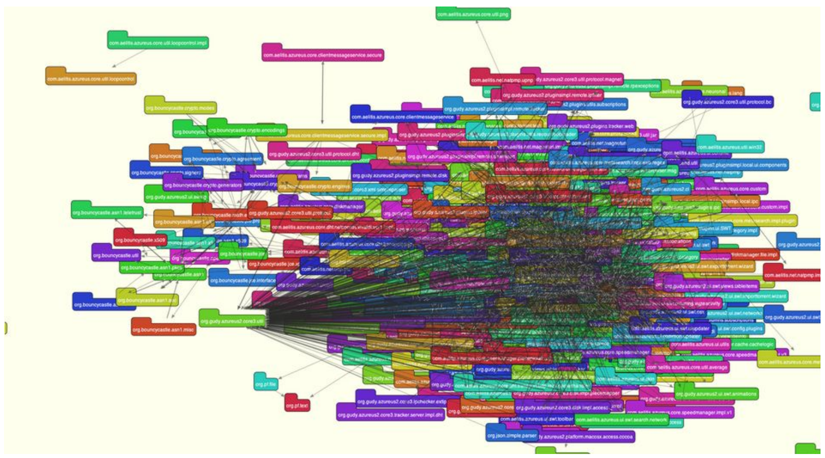
Increasing Complexity



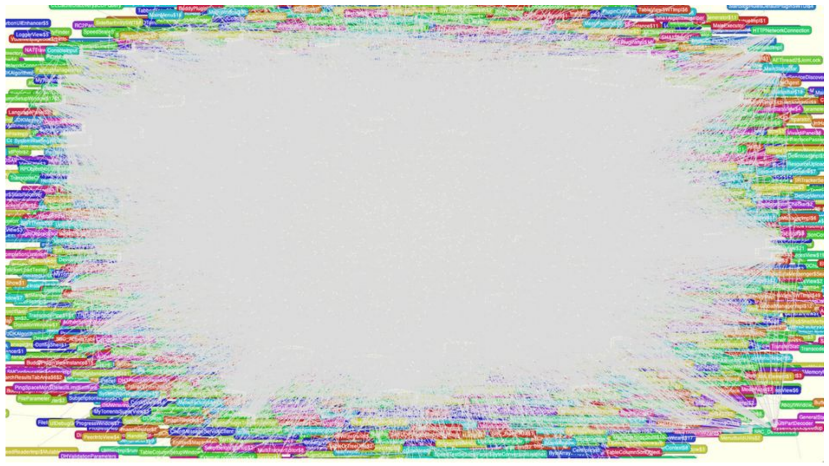
- four versions of ANTLR - 2.7.0 - 3.2
- ANTLR is a popular Java program used to define languages and generate parsers
- the graphs show ANTLR packages and their relationships
- complexity: increasing number of elements and interactions/relationships between them

Too Much Complexity

Software can become so complex that it becomes completely unmanageable.



java packages and their relationships in azureus



java classes and their relationships in azureus

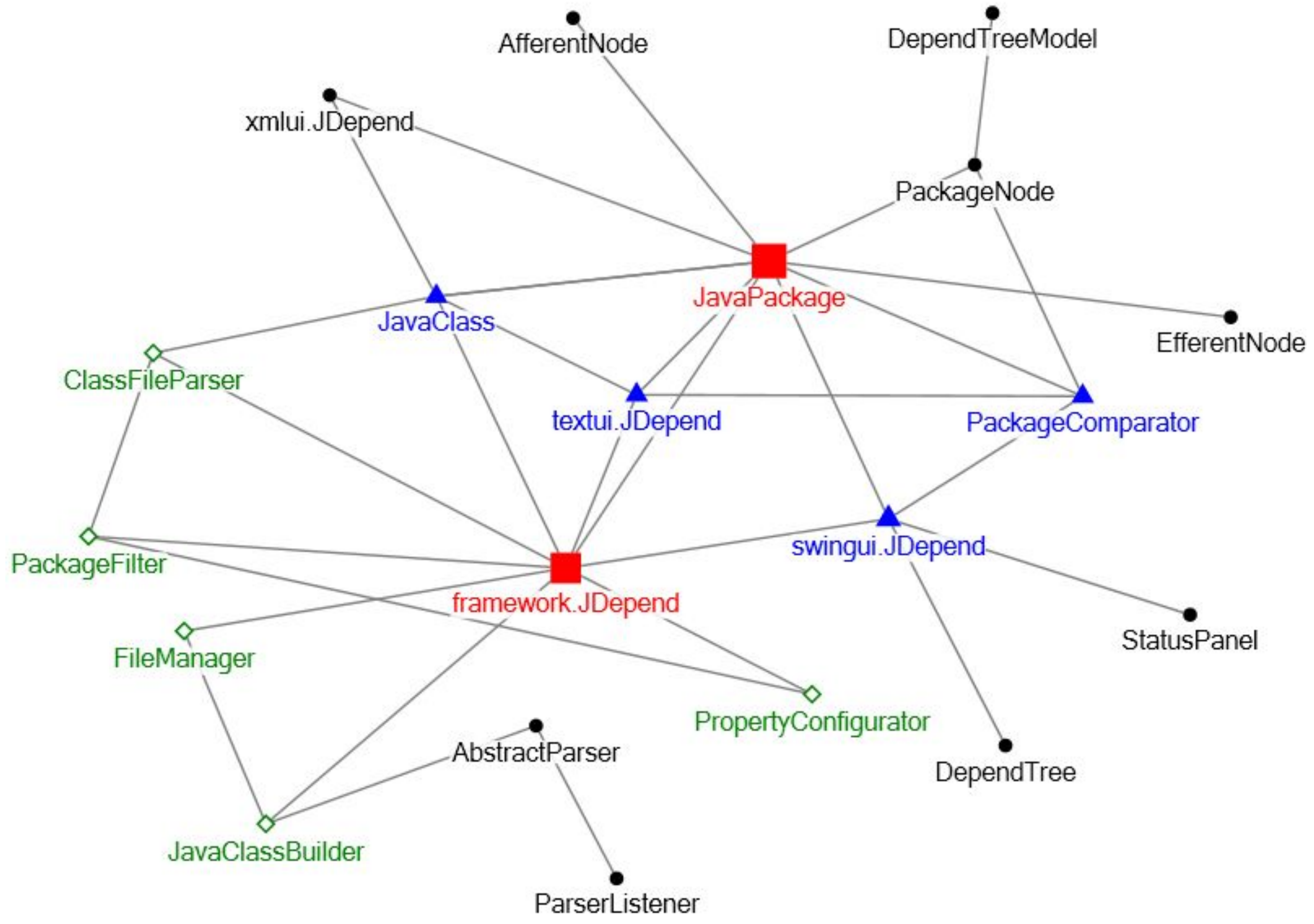
This is sometimes called **big ball of mud**.

Big Ball of Mud

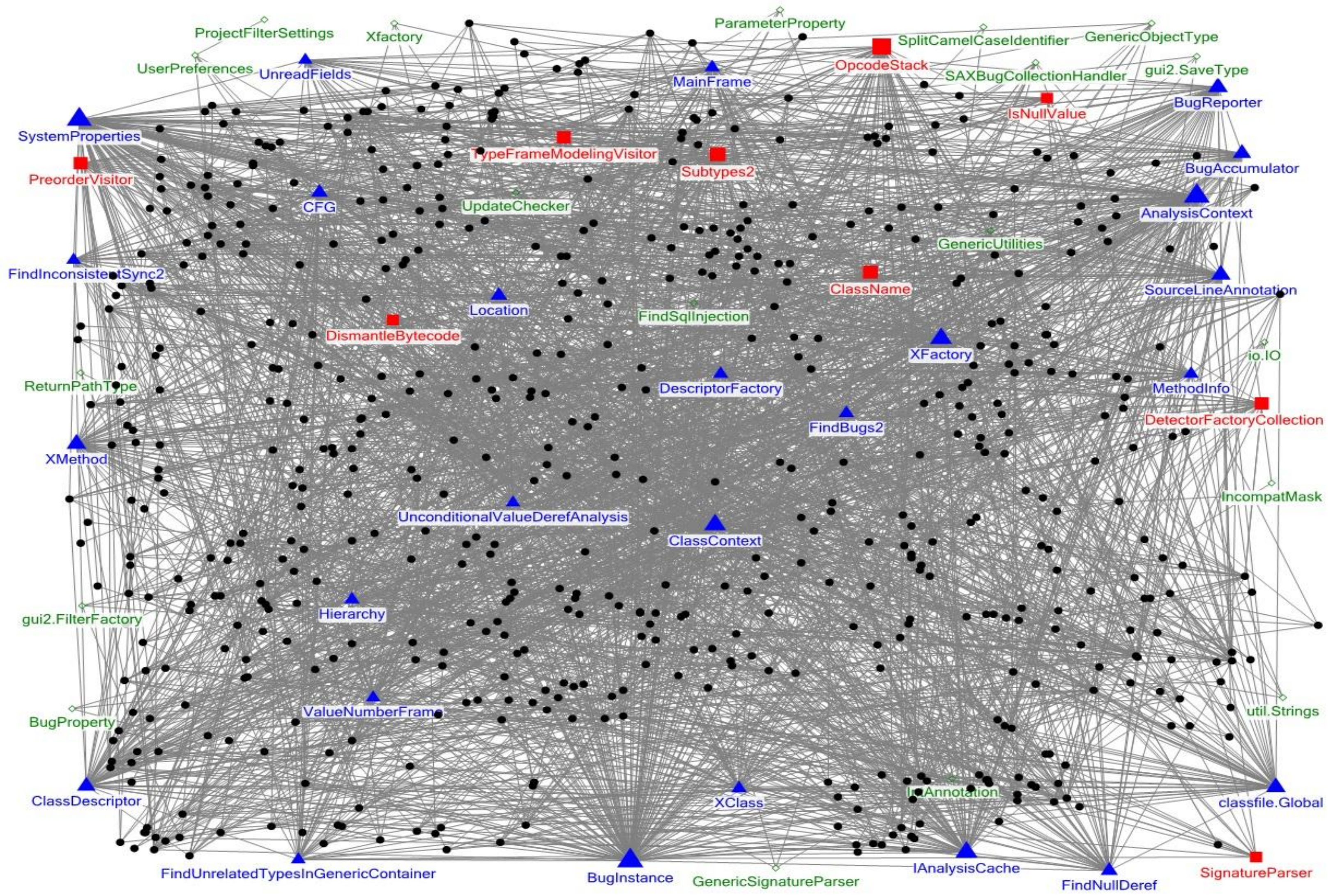
A BIG BALL OF MUD is haphazardly structured, sprawling, sloppy, DuctTape and bailing wire, **SpaghettiCode jungle**. .. These systems show unmistakable signs of **unregulated growth**, and repeated, expedient repair. Information is **shared** promiscuously **among distant elements** of the system**,** often to the point where nearly all the important information becomes **global** or **duplicated**. The overall structure of the system may never have been well defined. If it was, it may have **eroded beyond recognition**.

B. Foote, J. Yoder 99.
http://www.laputan.org/mud/mud.html#BigBallOfMud

Dynamic dependencies (JDepend)
(when you run the program)



Dynamic dependencies (FindBugs/SpotBugs)
(when you run the program)



Why Does Software Need to Change?

change pressure from:

- maintenance - bugfixes etc
- adapt to changing environment (example: Y2K !)
- implement new requirements
- add bells and whistles

Lehmann's Law

1. The law of **continuing change**. Any software system used in the real-world **must change** or become less and less useful in that environment.
2. The law of **increasing complexity**. As time flows forwards, **entropy increases**. That is, as a program evolves, its structure will **become more complex**.

source: Lehman, M.: Programs, life cycles and the laws of software evolution," *Proc. IEEE*, 15 (3), 1980.

The World is Complex !

Change Pressure: Y2K

- unavoidable change pressure came from the Gregorian Calendar in the late 90ties
- problem faced by computers to roll over 2 digit dates to 4 digits dates (from 1999 to 2000, and not to 1900)
- old software had built (wrong) assumptions over the lifetime of the system into the system
- estimated cost to fix this: "Worldwide, organizations were estimated to have spent **$308 billion** before the millennium on remediation efforts."
  source: Computerword.
- some good news: a Y2K movie was also made

The Formula One Dilemma

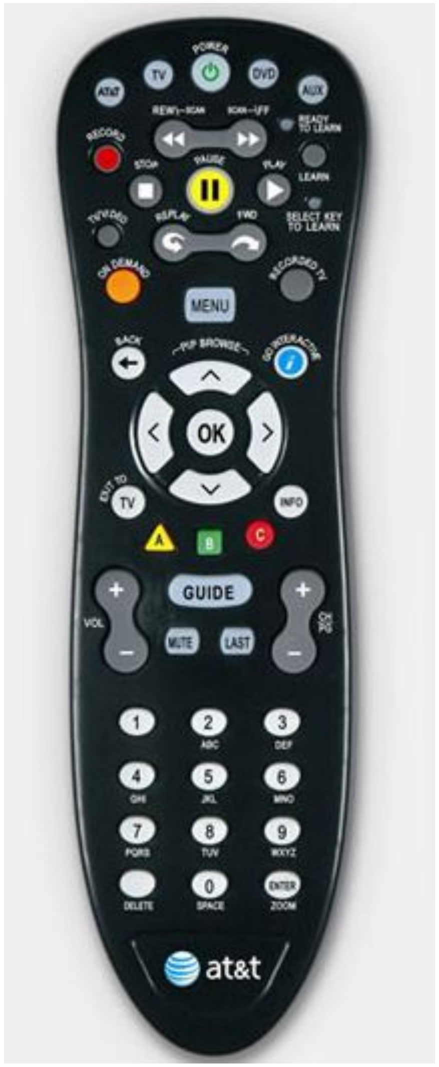
https://www.youtube.com/watch?v=3V1ZjbKGgR0

- Cars are now much dependent on Software.
- Each bug may contribute to the win/loss of a race and also to the safety of the driver.
- With a short timeframe between races, bugs need to be fixed and software need to be retested.
- After each race, engineers sit together and work hard to analyse bugs data and try to fix critical bugs before the next race.

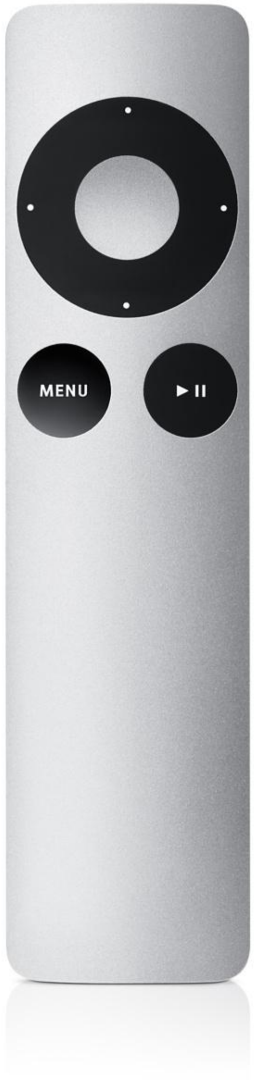
Feature Creep!

- MS Word 2003 has **292** menu items that needed to be mapped to Word 2007 (source)
- this is a sign of **feature creep** - the inclusion of non-essential feature (aka Creeping Featuritis)
- programmers confuse **what is needed** and what is **neat**
- this is sometimes also called "everything but the kitchen sink" syndrome
- reasons:
  - programmers with ego
  - marketing departments
  - (perceived) competitive advantage: based on the idea that value = number of features

Feature Creep ctd



VS



YAGNI & co

this observation has lead to the following (agile) principles:

- **YAGNI** - you are not going to need it - always implement things when you **actually** need them, never when you just **foresee** that you need them
- **KISS** - keep it simple and straightforward
- **use case - driven development** - the development *should be driven by customer requirements* represented by use cases - core principle of agile se

The Quest for Simplicity

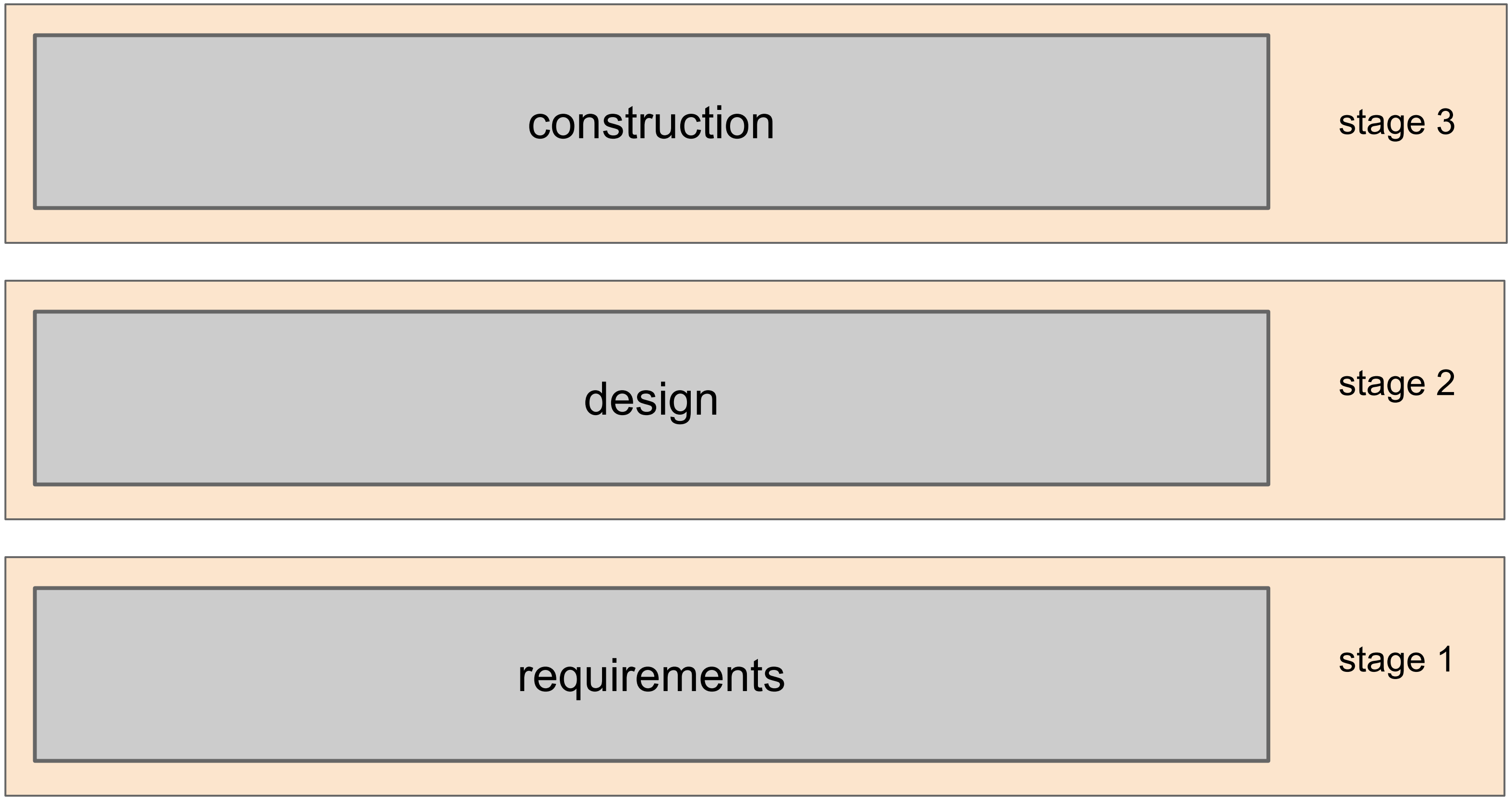
- design driven by actual requirements
- the simplest design is the best
- functionality first
- complexity must be avoided

vs

“Occam’ razor” - William of Ockham (ca 1300)
“Less is More” - Ludwig Mies van der Rohe (ca 1960)
Agile Software Engineering (from ca 1995)

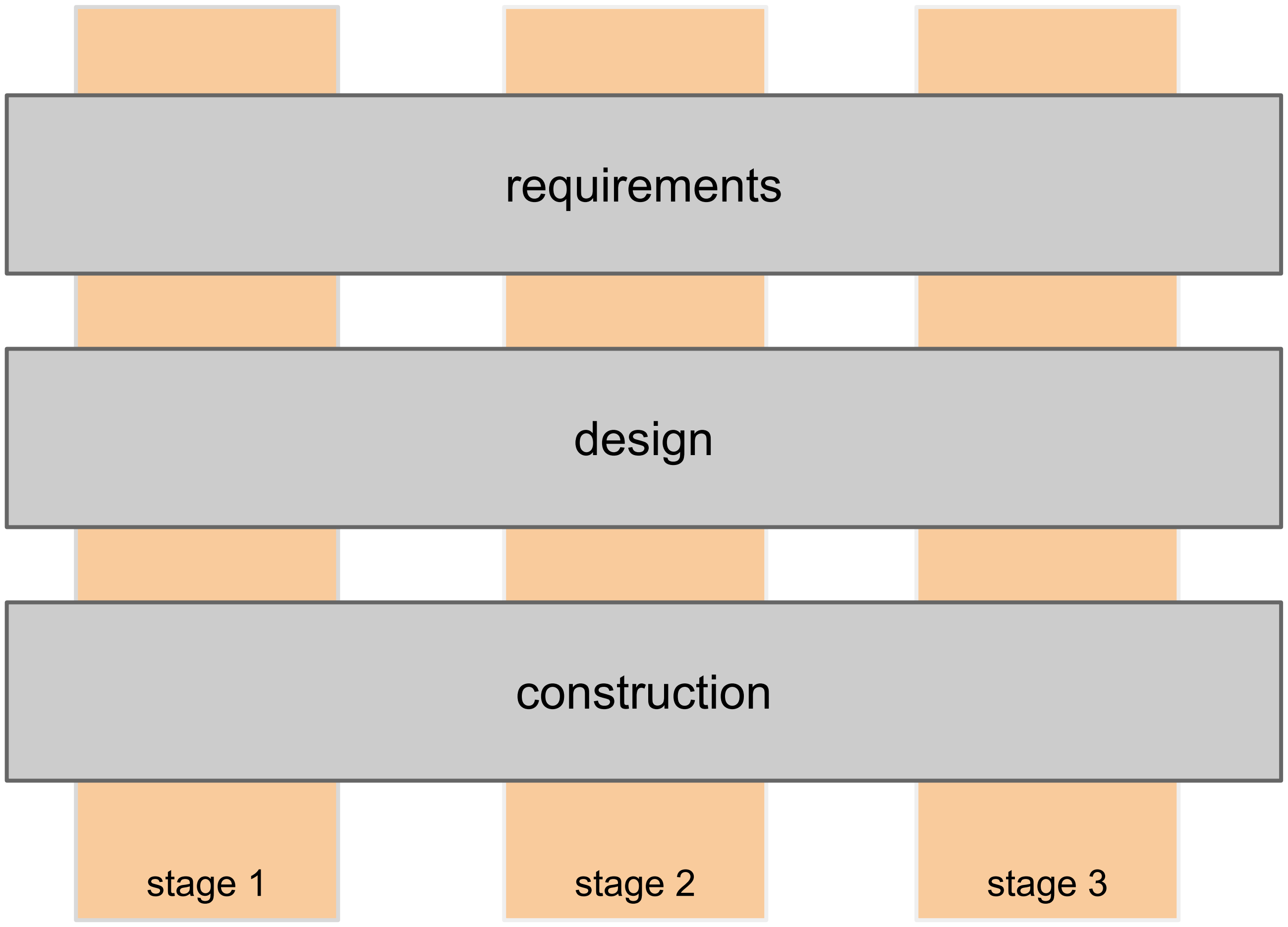
What to Implement

traditional (waterfall): gather all requirements, big upfront design, implement everything, test & deliver

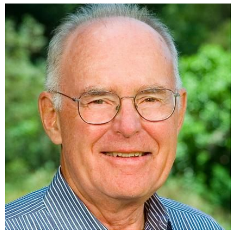


What to Implement

iterative / agile: implement one requirement at a time, then build and deliver



Moore's law



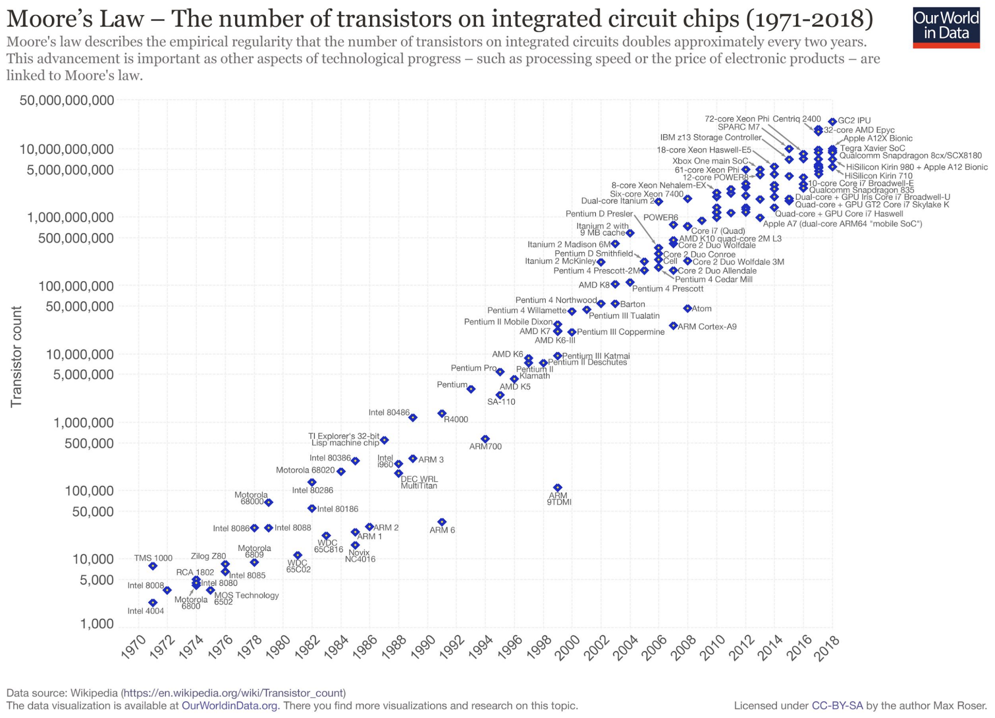
In 1965, Gordon E. Moore predicted that the number of transistors in a dense integrated circuit doubles about every two years.

Software had to keep up with the fast growth in hardware.

Note: Moore’s law is virtually broken as the high temperatures of transistors eventually would make it impossible to create smaller circuits (some expects it to completely break in the mid 2020s)
https://www.investopedia.com/terms/m/mooreslaw.asp

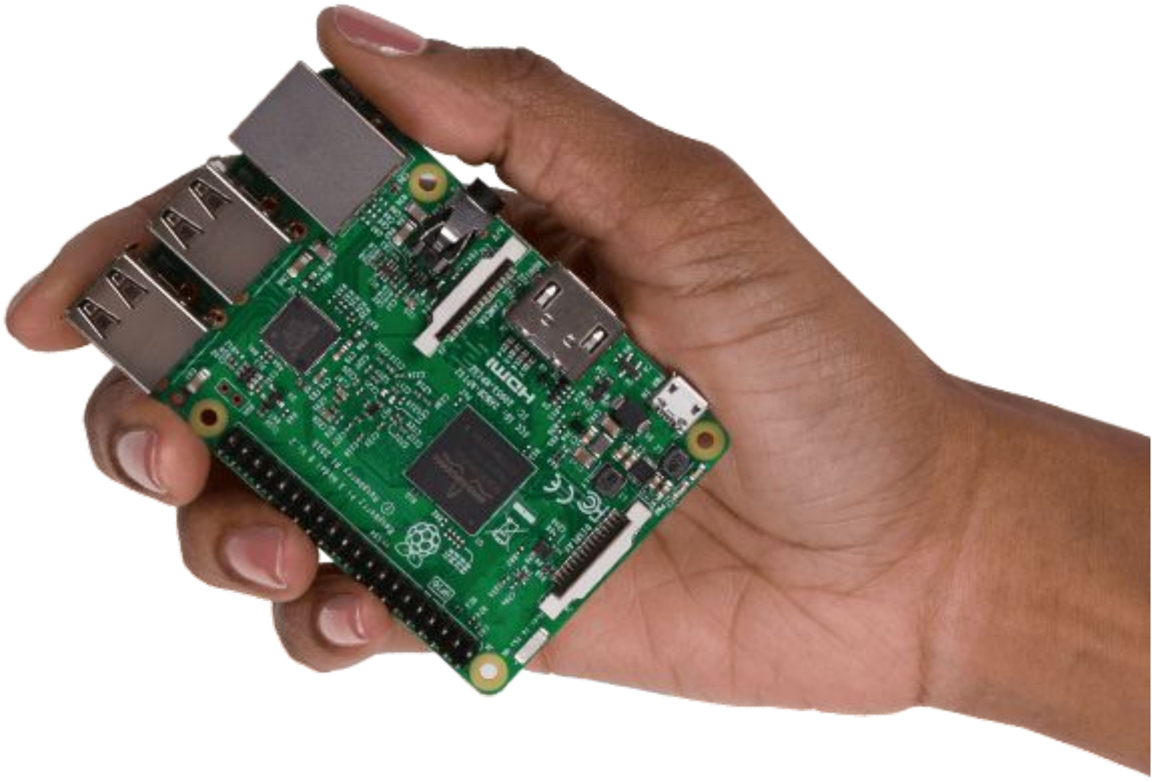
Watch out for Quantum Physics/Computing.

If interested, have a look at Moore's second law (aka Rock's law).



Not all software require large hardware!

- Programming with Raspberry pi!



Cost less than $20!
Quad-core, Arm Cortex A53 processor
4K video playback $x full HD (1080p),
1GB DDR3

HDMI 2.0a
Gigabit Ethernet.
2 USB3 ports

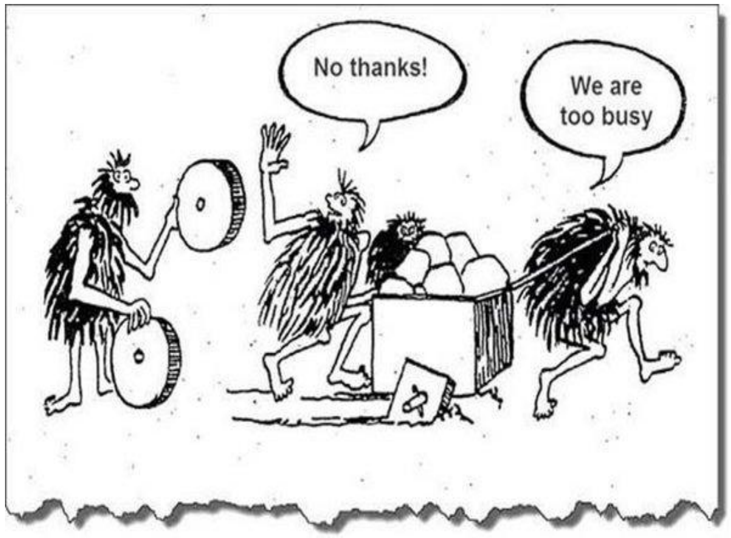
Weight 50G approx.

Software Quality Attributes

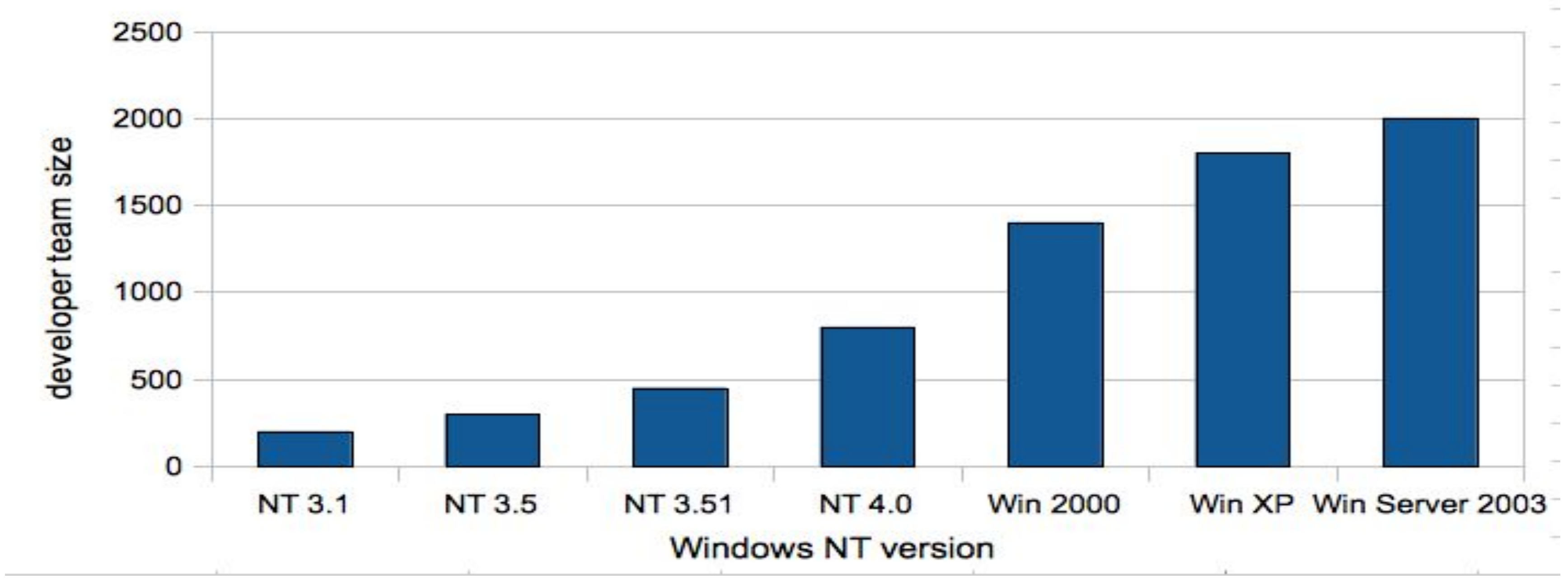
- but what is good software?
- define quality attributes, and how to measure them
- choices:
  - observe projects, and use statistics - requires long investment
  - use **metrics** to measure quality attributes on system

Tradeoffs

- quality is desirable, but so is time to market
- the extreme version of this is "quick and dirty"
- often, tradeoffs must be made
- then it becomes important to **document** and manage **issues**
- technical debt is a metaphor used to describe this (proposed by W. Cunnigham)
- note that **technical debt incurs interest payment**!



Increasing Team Sizes



- the size of Windows NT developer teams from 1993-2003
- source: V. Maraia: The Build Master: Microsoft's Software Configuration Management Best Practices. Addison-Wesley 2005.

Managing (Distributed) Teams

- use central (server-based) **revision control systems**
- aka (code) **repositories**
- manage code versions + multiple programmers collaborating
- revert to previous versions, branch and merge
- manage concurrent modifications
- example: GIT, SVN, Mercurial, CVS

Outsourcing

- trend that big organisations outsource work to constructors
- makes businesses more flexible
- often, work is outsourced to low cost countries like India (aka **offshoring**)
- this results in large scale, distributed development that is often difficult to manage

Programmer Productivity

What is better - many low-skilled or few high-skilled programmers?

Programming managers have long recognized wide productivity variations between good programmers and poor ones. But the actual measured magnitudes have astounded all of us. In one of their studies, Sackman, Erickson, and Grant were measuring performance of a group of **experienced** programmers. Within just this group the ratios between the best and worst performances averaged about **10:1 on productivity measurements ..**

F. Brooks: The Mythical Man-Month: Essays on Software Engineering

Addison Wesley 1995

note: the Sackman study is from the 68 (http://goo.gl/n31ZL)

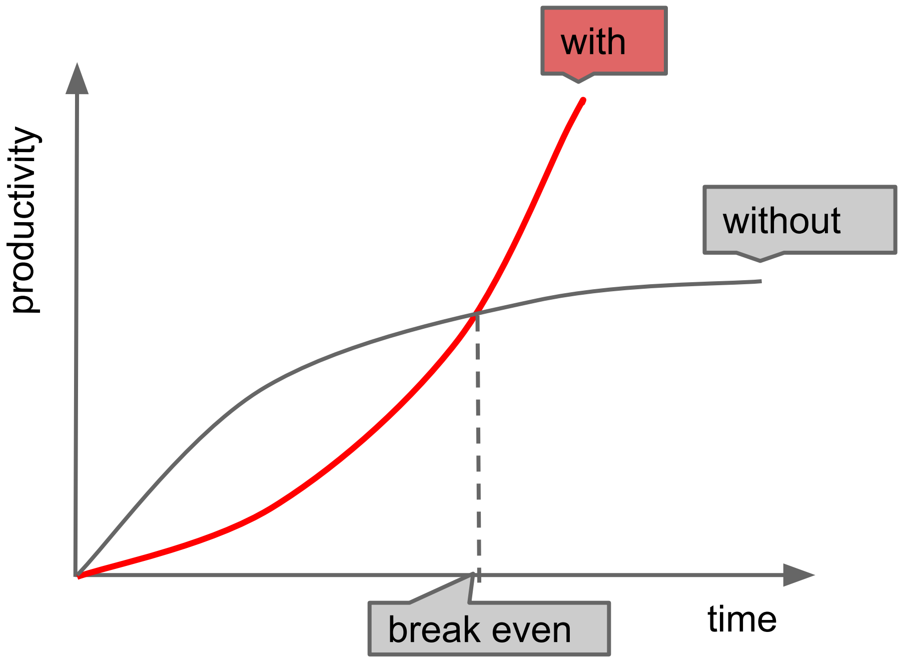
Programmer Productivity (ctd)

The best programmers are up to **28 times better** than the worst programmers, according to “individual differences” research. **Given that their pay is never commensurate, they are the biggest bargains in the software field.**

R. Glass: Facts and Fallacies of Software Engineering. Addison-Wesley 2003.

Is it worth the effort?

- the tools and methods discussed have a learning curve: there is no free lunch
- however, we selected T+M s for which we think that the **break even point** occurs **quickly** even for small projects



Scope of this Paper

1. **managing quality**
2. **working in teams**
3. **managing change**

159.251

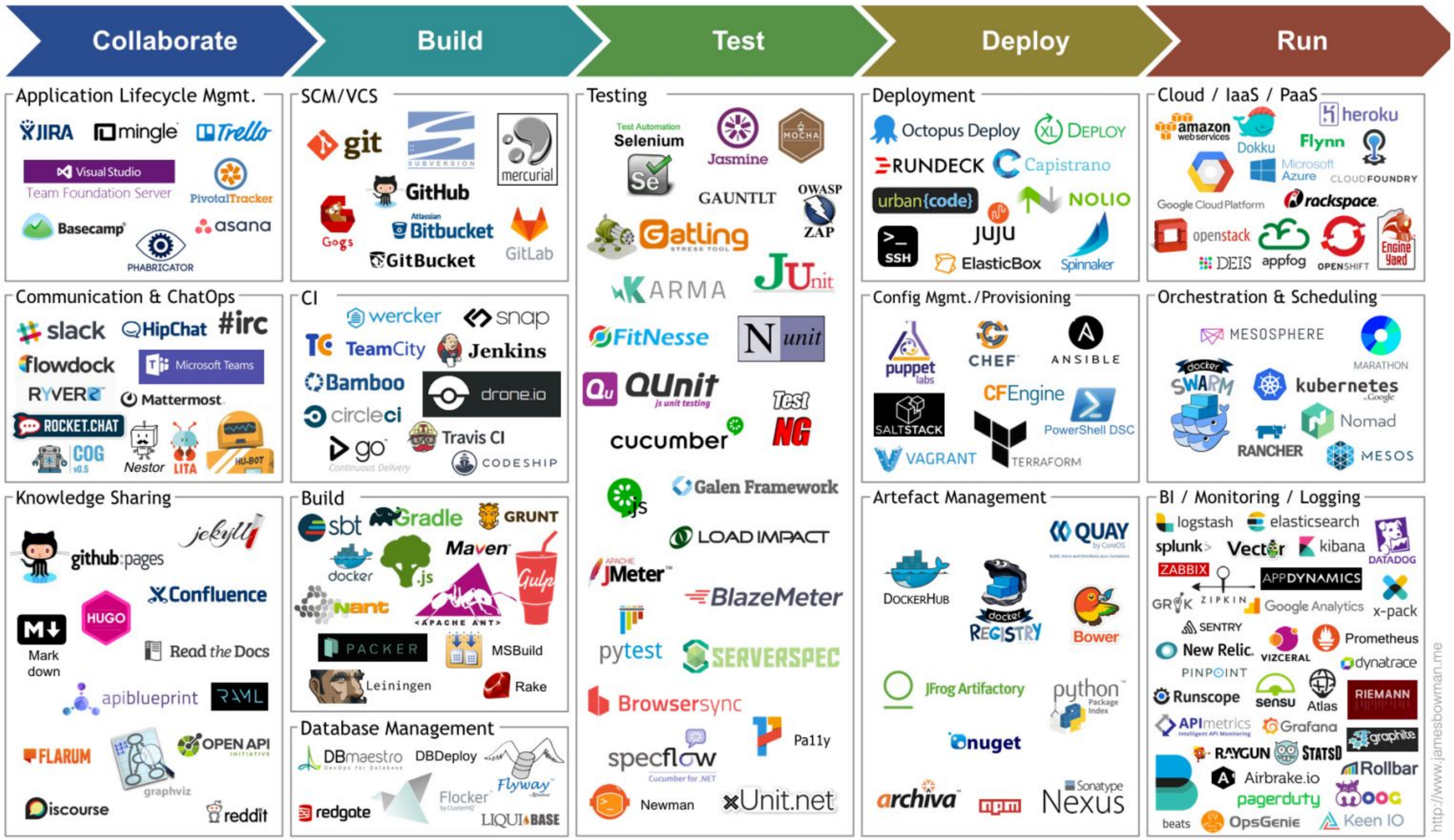
4. managing scope
5. planning
6. estimating

158.225

Tools and Methods

In this paper, we will discuss tools and methods to:

1. improve the productivity of a software engineer
2. improve the quality of the software built


Collaborate
Build
Test
Deploy
Run
Application Lifecycle Mgmt.
JIRA
mingle
Trello
Visual Studio Team Foundation Server
PivotalTracker
Basecamp
asana
PHABRICATOR
Communication & ChatOps
slack
HipChat
#irc
flowdock
Microsoft Teams
RYVER
Mattermost
ROCKET.CHAT
COG v0.5
Nestor
LITA
HU-BOT
Knowledge Sharing
github:pages
jekyll
Confluence
HUGO
Mark down
Read the Docs
apiblueprint
RAML
FLARUM
graphviz
OPEN API INITIATIVE
Discourse
reddit
SCM/VCS
git
SUBVERSION
mercurial
GitHub
Gogs
Atlassian Bitbucket
GitLab
GitBucket
CI
wercker
snap
TeamCity
Jenkins
Bamboo
drone.io
circleci
Travis CI
go Continuous Delivery
CODESHIP
Build
sbt
Gradle
GRUNT
Maven
docker
.js
Gulp
Nant
APACHE ANT
PACKER
MSBuild
Leiningen
Rake
Database Management
DBmaestro DevOps for Database
DBDeploy
Flyway
Flocker by ClusterHQ
redgate
LIQUIBASE
Testing
Test Automation Selenium
Se
Jasmine
MOCHA
GAUNTLT
OWASP ZAP
Gatling STRESS TOOL
KARMA
JUnit
FitNesse
NUnit
QUnit js unit testing
TestNG
cucumber
.js
Galen Framework
LOAD IMPACT
APACHE JMeter
BlazeMeter
pytest
SERVERSPEC
Browsersync
Pa11y
specflow Cucumber for .NET
Newman
xUnit.net
Deployment
Octopus Deploy
XL DEPLOY
RUNDECK
Capistrano
urban{code}
NOLIO
JUJU
SSH
ElasticBox
Spinnaker
Config Mgmt./Provisioning
puppet labs
CHEF
ANSIBLE
CFEngine
SALTSTACK
PowerShell DSC
VAGRANT
TERRAFORM
Artefact Management
QUAY by CoreOS
DockerHub
docker REGISTRY
Bower
JFrog Artifactory
python Package Index
nuget
archiva
npm
Sonatype Nexus
Cloud / IaaS / PaaS
amazon webservices
Dokku
heroku
Flynn
Microsoft Azure
CLOUDFOUNDRY
Google Cloud Platform
rackspace
openstack
DEIS
appfog
OPENSHIFT
Engine Yard
Orchestration & Scheduling
MESOSPHERE
MARATHON
docker SWARM
kubernetes by Google
Nomad
RANCHER
MESOS
BI / Monitoring / Logging
logstash
elasticsearch
splunk>
Vector
kibana
DATADOG
ZABBIX
APPDYNAMICS
GROK
ZIPKIN
Google Analytics
x-pack
SENTRY
New Relic.
VIZCERAL
Prometheus
PINPOINT
dynatrace
Runscope
sensu
Atlas
RIEMANN
APImetrics
Grafana
graphite
RAYGUN
STATSD
Rollbar
Airbrake.io
pagerduty
beats
OpsGenie
Keen IO
http://www.jamesbowman.me