Technical Excellence in Software Development

Prof. Ingrid Nunes
Universidade Federal do Rio Grande do Sul (UFRGS)

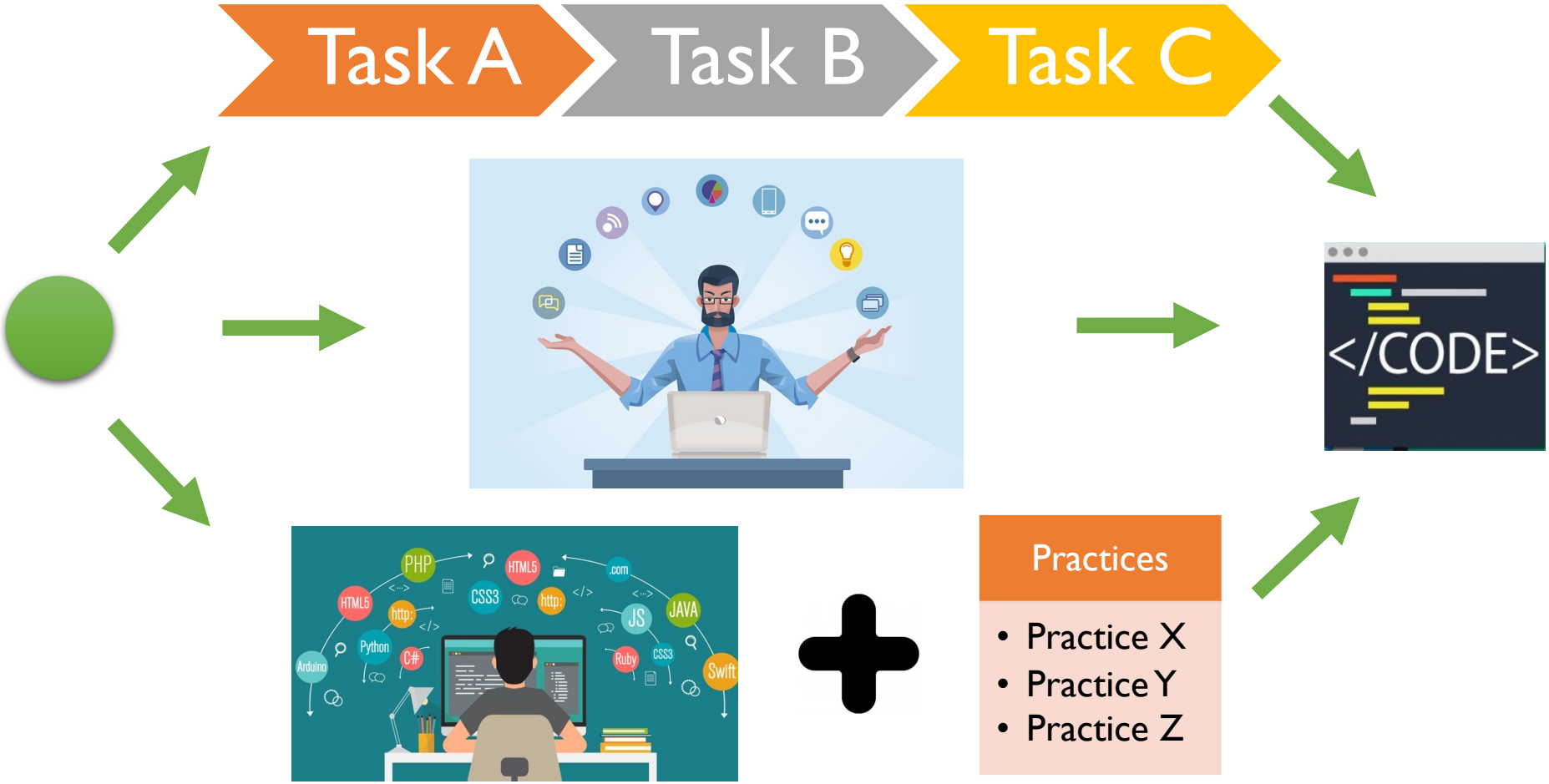
Série de Seminários do INF + CEI-Talks

INSTITUTO DE INFORMÁTICA UFRGS

Prosoft

UFRGS UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL

Software Development



R. L. Glass, "Greece vs. Rome: Two Very Different Software Cultures," in IEEE Software, vol. 23, no. 6, pp. 112-112, Nov.-Dec. 2006.doi: 10.1109/MS.2006.163

Software Development

- But…

Software Development

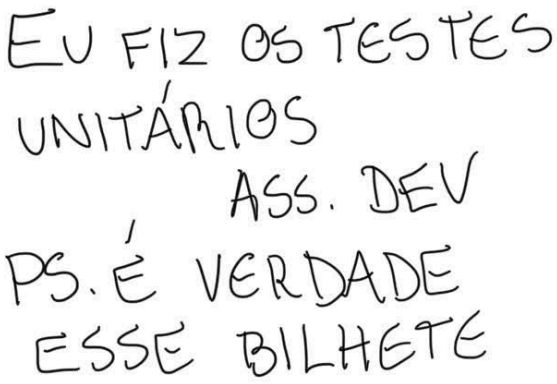
- And…

https://the-stack-overflow-podcast.simplecast.com/episodes/code-base-clean-modern-roberta-arcoverde

Software Development

- What does not work?

1. To adopt an approach without knowing it or adapt it, when it is required
2. To adopt an approach ignoring what is key for it to work
3. To consider something as does with no guarantees that it has been done as it should be



Principles and Best Practices

```
class UserController {

    createUser() {
        // check data valid
        new User()
    }

    resetPassword() {
        // check token valid
        // check pw valid
        user.setPassword(newPW)
    }

}
```

```
class User {

    login
    password

    //getters

    //setters

}
```

Is there any problem in this code?

Principles and Best Practices

```
class UserController {

    createUser() {
        // check data valid
        new User()
    }

    resetPassword() {
        // check token valid
        // check pw valid
        user.setPassword(newPW)
    }

}
```

```
class User {

    login
    password

    //getters

    //setters

}
```

Duplicated code!

High coupling!

Principles and Best Practices

```
class UserController {

    createUser() {
        checkUser()
        new User()
    }

    resetPassword() {
        // check token valid
        chec    r()
    }
}
```

```
class User {

    login
    password

    //getters

    //setters

}
```

```
class UserValidator {
    checkUser() { … }
}
```

Procedural Paradigm ???

Principles and Best Practices

- Effective use of object orientation

```
class UserController {

    createUser() {
        new User()
    }

    resetPassword() {
        user.resetPassword(newPW)
    }

}
```

```
class User {

    login
    password

    User () {
        ...
        setPassword()
    }
    setPassword() {
        //check pw valid
    }
    resetPassword() {
         isTokenValid()
         setPassword()
    }
    isTokenValid() { … }
}
```

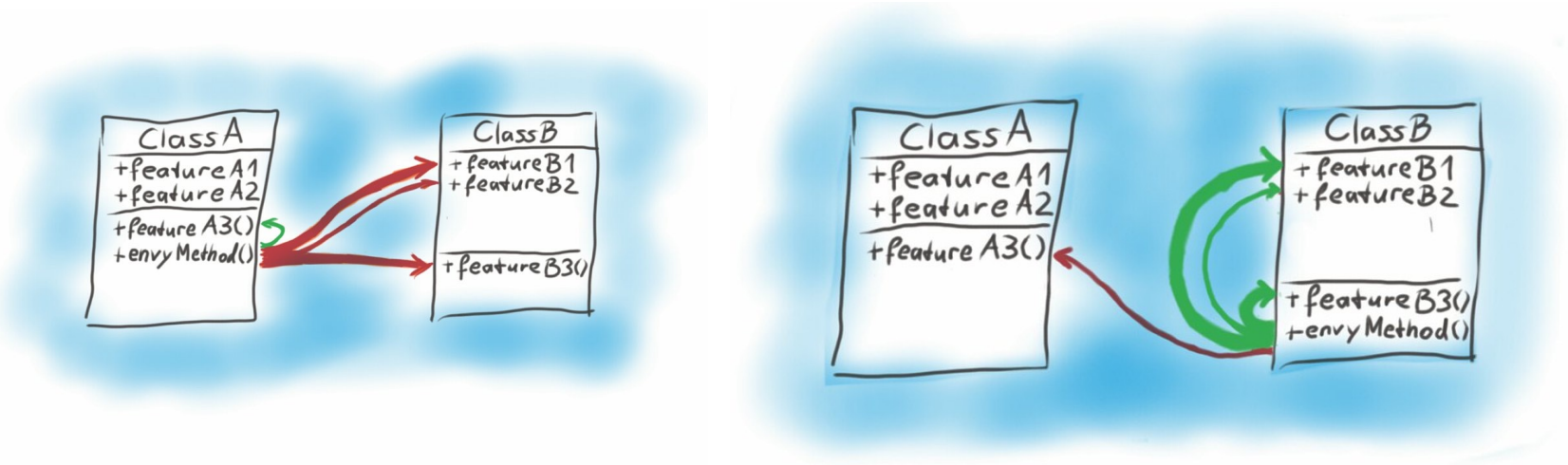
Principles and Best Practices

- Rules and Principles
  - Bertrand Meyer. 1988. Object-Oriented Software Construction.
  - Modularity Rules
    - Direct Mapping
    - Few Interfaces
    - Small interfaces (weak coupling)
    - Explicit Interfaces
    - Information Hiding
  - Modularity Principles
    - Linguistic Modular Units
    - Self-Documentation
    - Uniform Access
    - Open-Closed
    - Single Choice

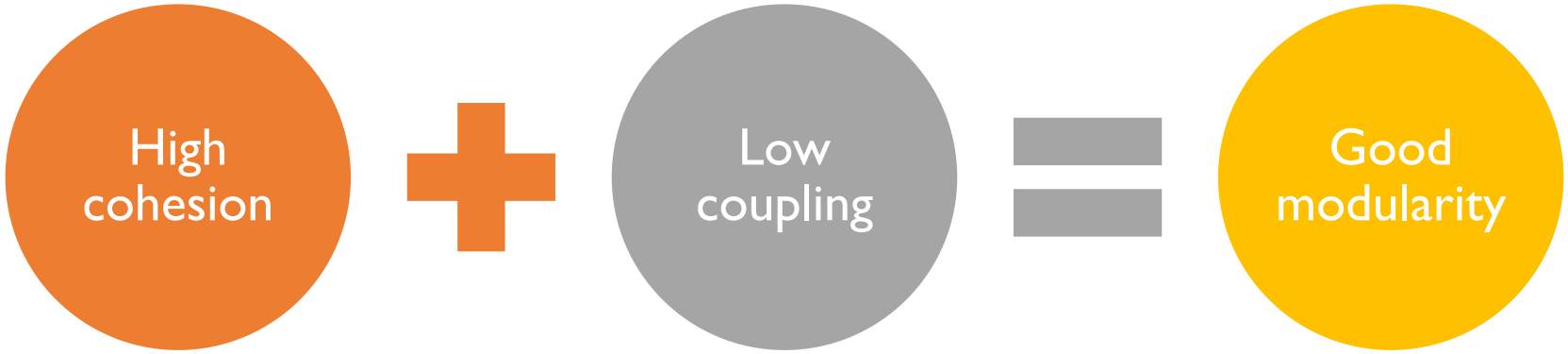
- Principles S.O.L.I.D.
  - Robert C. Martin. 2008. Clean Code: A Handbook of Agile Software Craftsmanship.
  - **S**ingle Responsibility Principles
  - **O**pen-Closed Principle
  - **L**iskov Substitution Principle
  - **I**nterface Segregation Principle
  - **D**ependency Inversion Principle

Principles and Best Practices

- GRASP Patterns
  - E.g. Specialist
  - Craig Larman. 1997. Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development.
- Code Smells / Refactoring Catalog
  - E.g.
    - Feature Envy
    - Data Class
  - Martin Fowler. 1999. Refactoring: Improving the Design of Existing Code.



Principles and Best Practices



Principles and Best Practices

On the Criteria To Be Used in Decomposing Systems into Modules

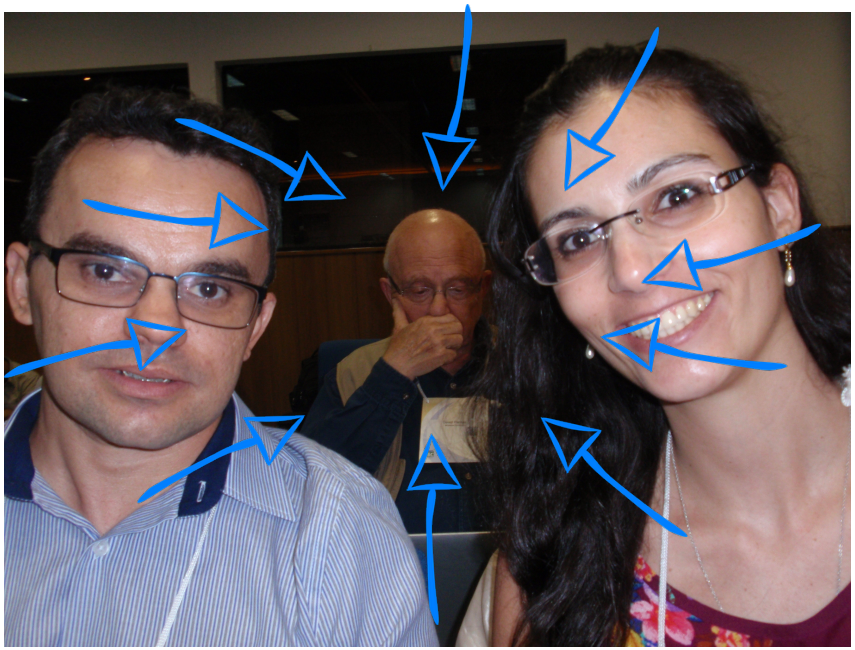
D.L. Parnas
Carnegie-Mellon University

A well-defined segmentation of the project effort ensures system modularity. Each task forms a separate, distinct program module. At implementation time each module and its inputs and outputs are well-defined, there is no confusion in the intended interface with other system modules. At checkout time the integrity of the module is tested independently; there are few scheduling problems in synchronizing the completion of several tasks before checkout can begin. Finally, the system is maintained in modular fashion; system errors and deficiencies can be traced to specific system modules, thus limiting the scope of detailed error searching.

Gouthier and Pont [1970]

Communications of the ACM — December 1972, Volume 15, Number 12

CBSoft 2014 – David Parnas

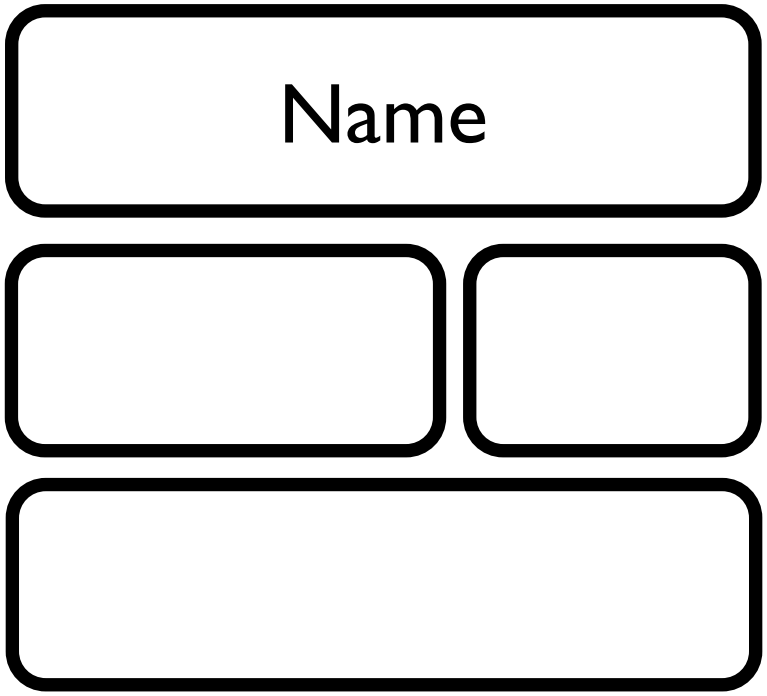


CBSOFT'21
congresso brasileiro de software
Mon 27 September - Fri 1 October

http://cbsoft2021.joinville.udesc.br/

Software Architecture

- Fundamental for an organised software evolution

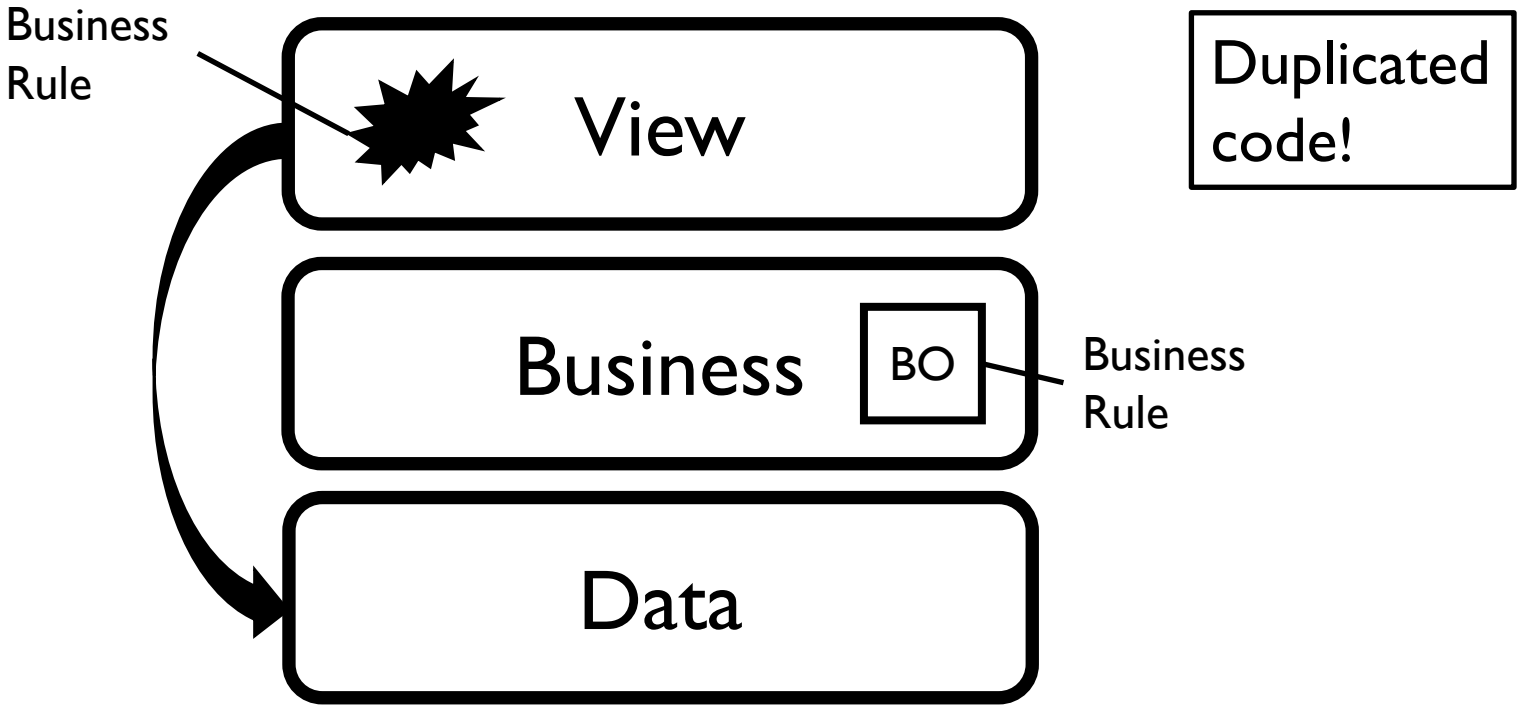


1. Modules
2. Dependencies
3. Module roles

Consistency between the conceptual and implemented architecture!

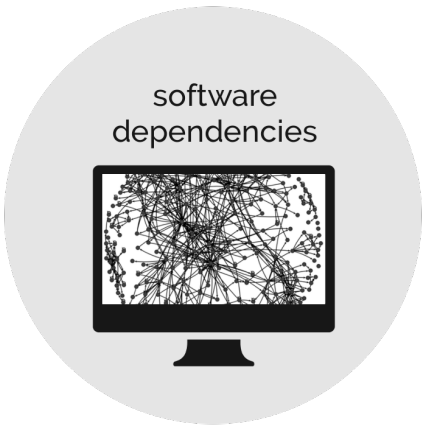
Software Architecture

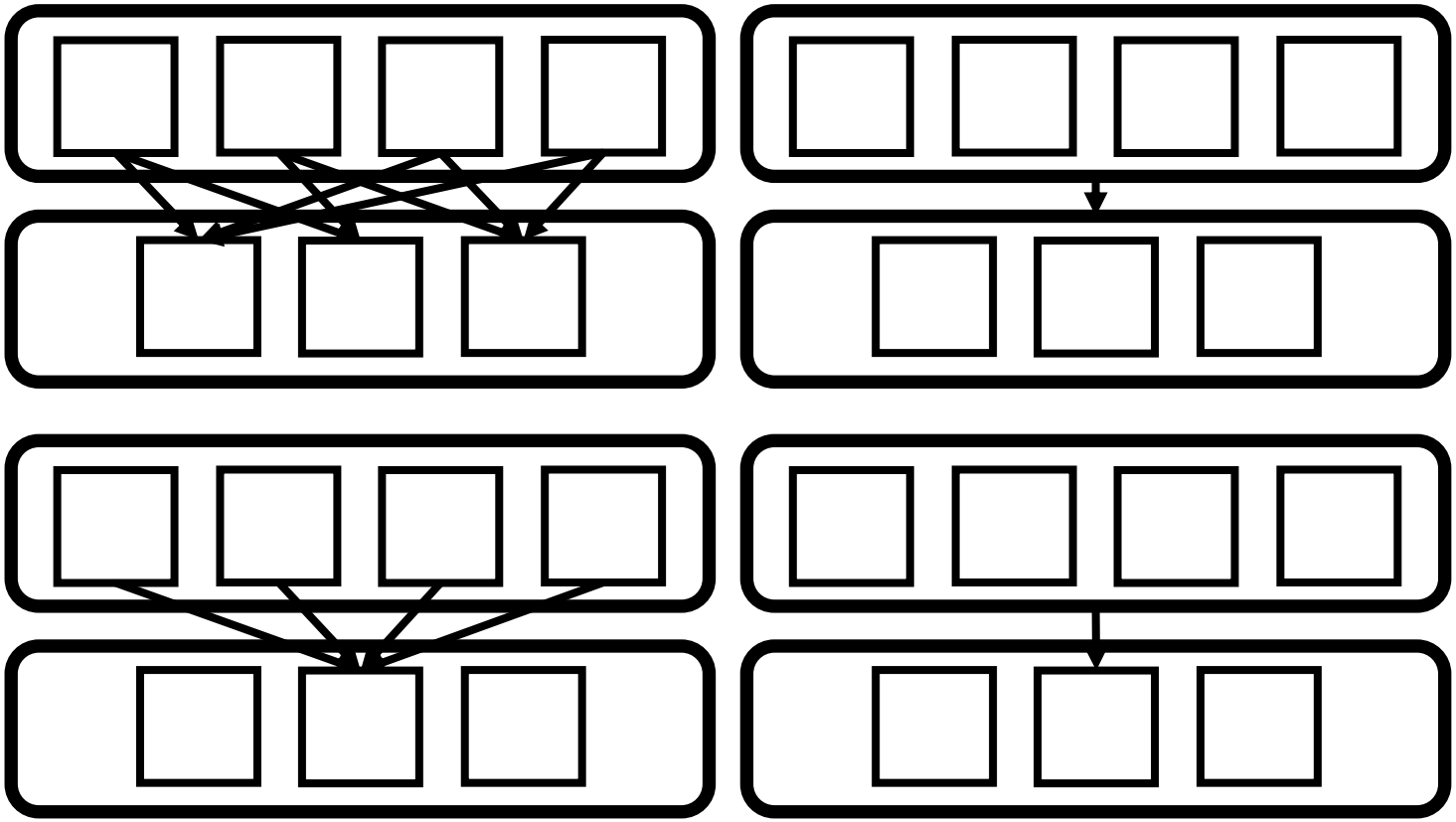
- Fundamental for an organised software evolution



Software Architecture

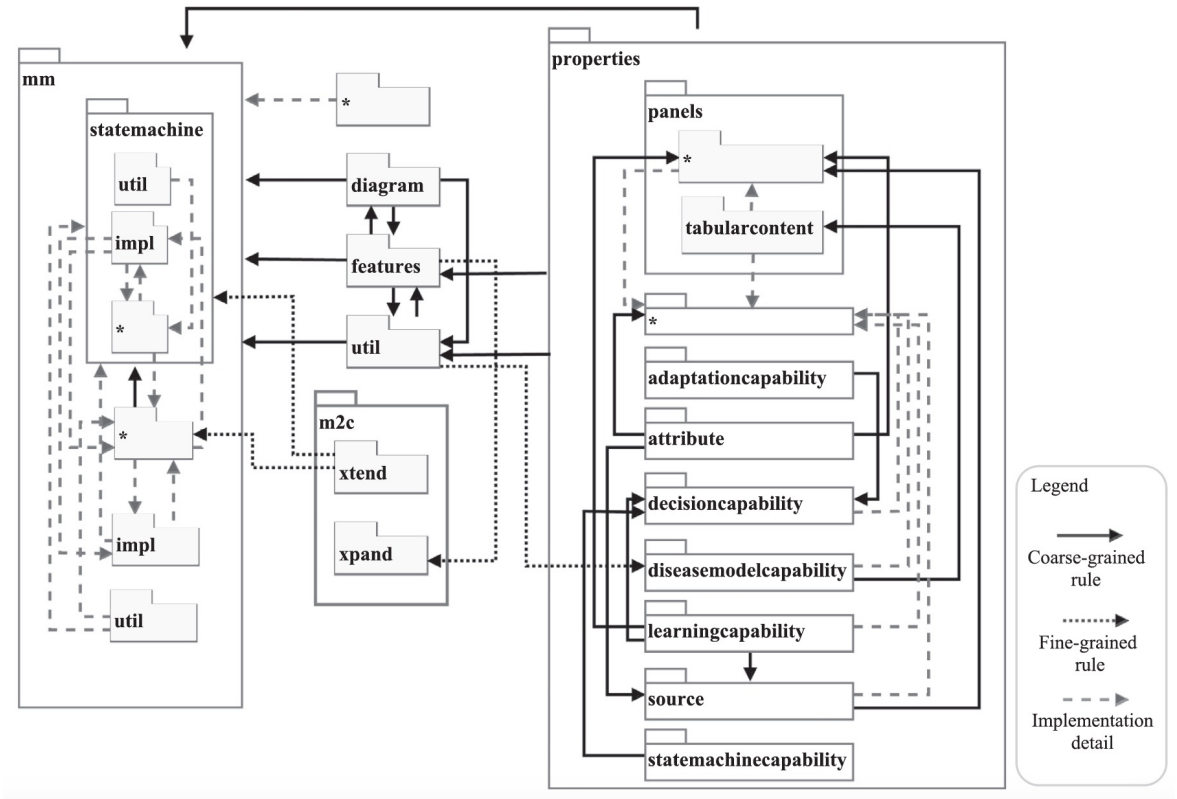
- Architecture Recovery





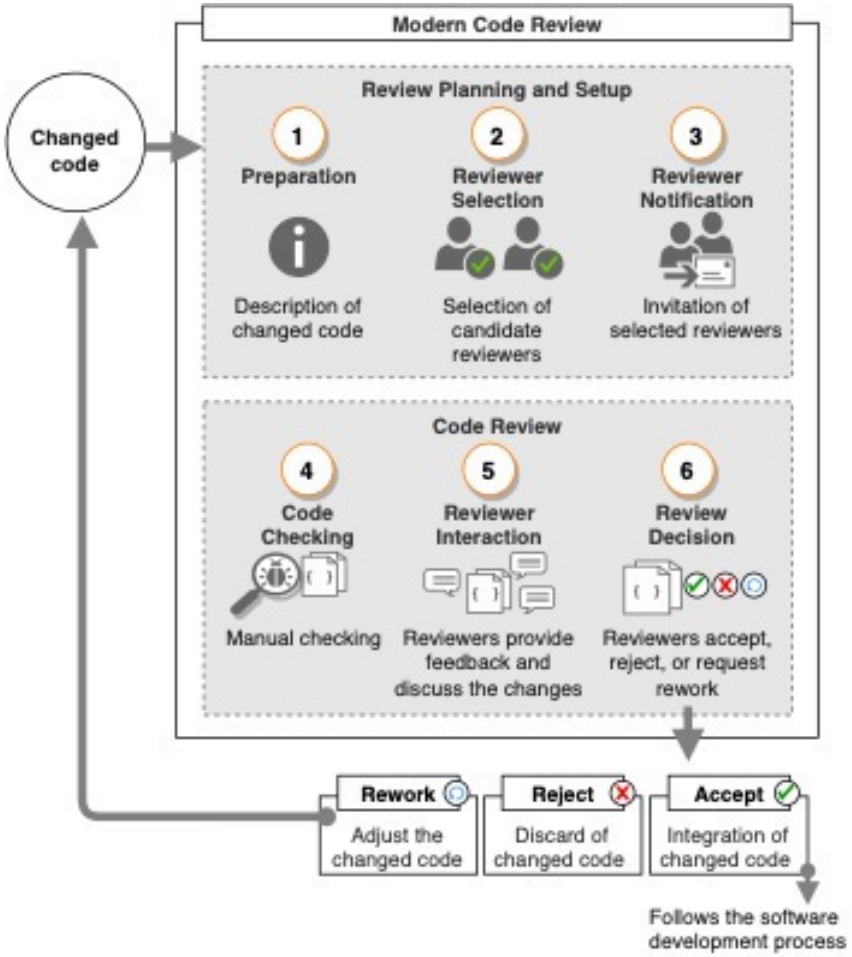
Software Architecture

- WGB Method
  - Dependencies
    +
    MDS Metric
    +
    Optimisation Problem



ZAPALOWSKI, V. ; NUNES, I. ; NUNES, D. . The WGB method to recover implemented architectural rules. INFORMATION AND SOFTWARE TECHNOLOGY, v. 103, p. 125-137, 2018.

Code Review



- Advantage
  - Avoid problem 3: To consider something as does with no guarantees that it has been done as it should be

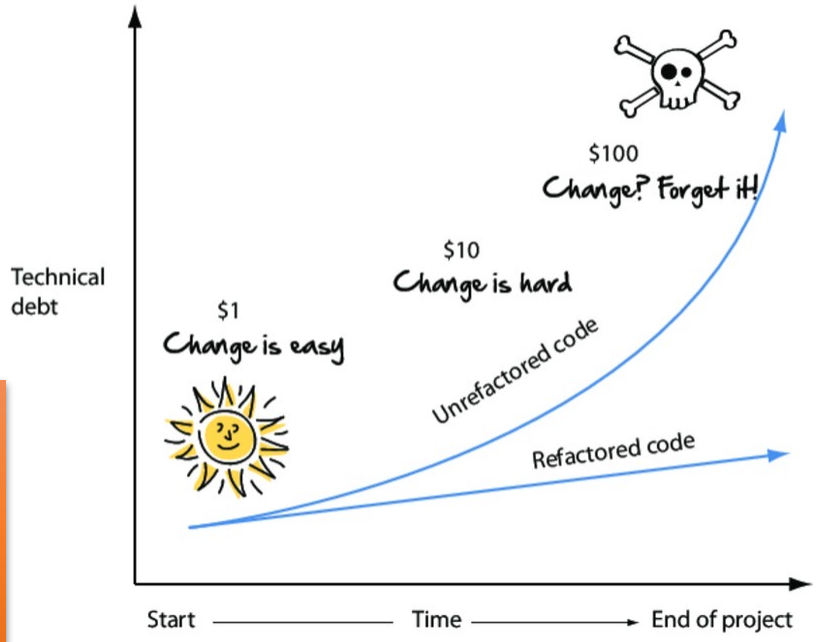
Code Review

- Some facts (most derived from OSS)
  - Average of 1-2 reviewers and 2-3 comments per request
  - Newcomers tend to receive more attention
  - Small code changes and long descriptions facilitate the review
  - Most frequent discussion topics
    - Code improvement
    - Understanding
    - Social interactions
- MCR and pair programming are interchangeable in terms of cost
  - When pair programming is adopted within test-driven development, MCR has lower cost
  - Unit testing: finds more failures
  - MCR: less time in the detection and isolation of the underlying sources of the defects
- Most common type of support
  - Reviewer recommenders and visualizations of code changes

DAVILA, N. ; NUNES, I. . A Systematic Literature Review and Taxonomy of Modern Code Review. The Journal of Systems & Software, 2021.

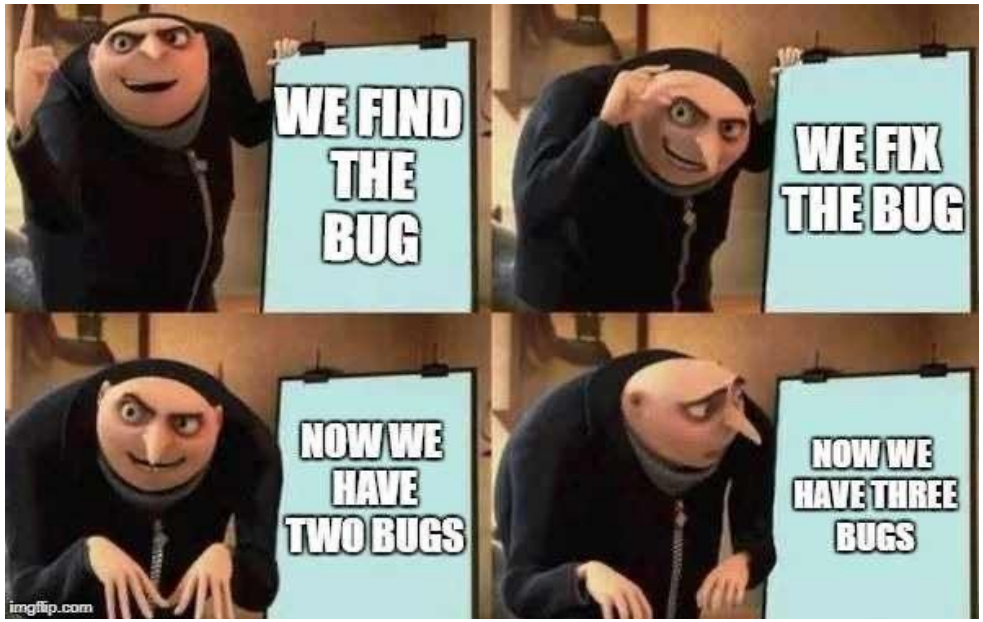
Technical Debt

- Metaphor created by Ward Cunningham to justify for non-technical stakeholders the need for refactoring
- Some problems in the code are like financial debt. It is ok to make a loan, as long as it is paid.



Technical debt management is crucial! It must be paid.

Technical Debt

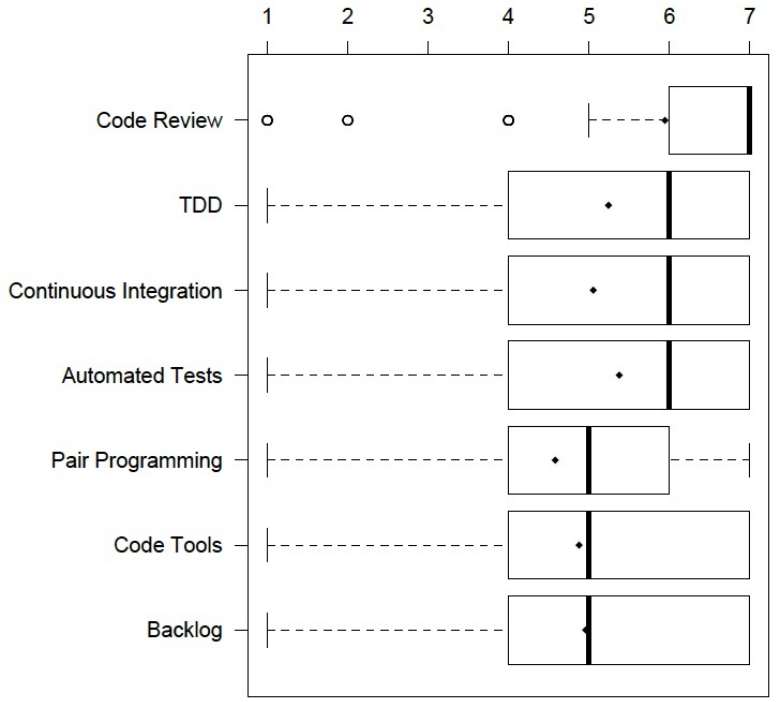


Technical Debt

Reasons for TD introduction at the code level

- From the self perspective
  - Tight schedule
  - Work overload
  - Pressure from the management
- From the perspective of other developers
  - Also development and technology inexperience

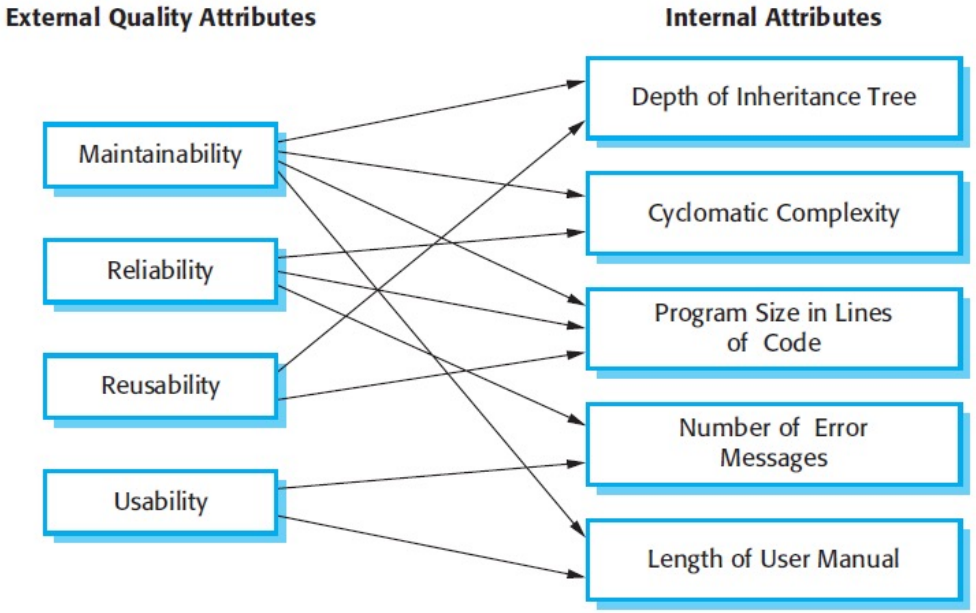
Practices that should be adopted to avoid TD



ROCHA, J. C. ; ZAPALOWSKI, V. ; NUNES, I. . Understanding Technical Debt at the Code Level from the Perspective of Software Developers. In: Brazilian Symposium on Software Engineering, 2017.

Metrics and Static Analysis Tools

- Code Metrics
  - Traditional: LOC, Fan-in, Fan-out, Cyclomatic Complexity, ...
  - CK Metrics: DIT, WMC, RFC, CBO, LCOM, NOC



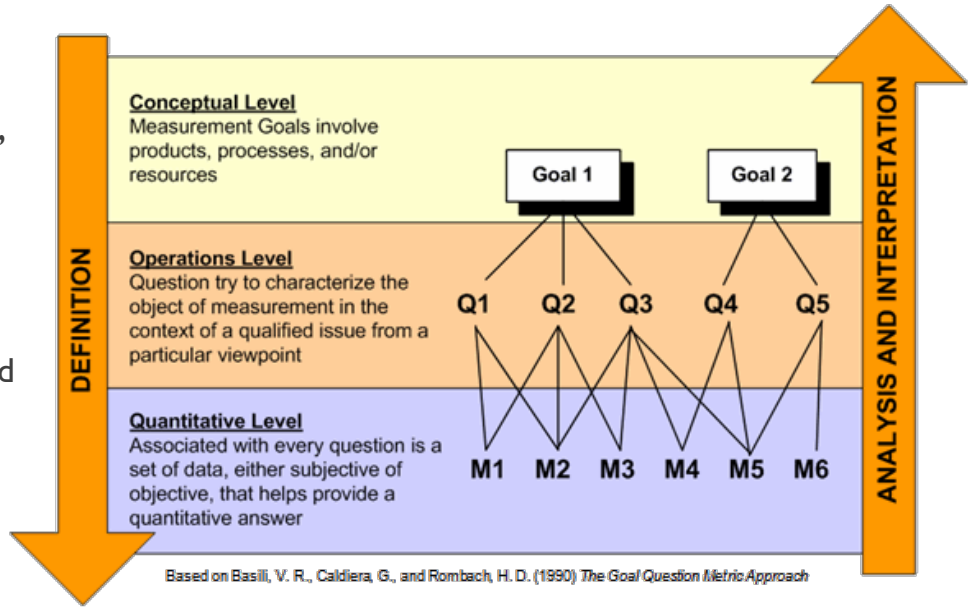
Metrics and Static Analysis Tools

- Code Metrics
  - Traditional: LOC, Fan-in, Fan-out, Cyclomatic Complexity, ...
  - CK Metrics: DIT, WMC, RFC, CBO, LCOM, NOC
- Static Analysis Tools
  - Automatically checking of common problems
  - Use of rules
    - Dependencies, code smell detections
  - Part of automated reviewers

Metrics and Static Analysis Tools

> "*You can't manage what you can't measure*",
>
> Tom DeMarco

- Experimental Software Engineering
- GQM
  - Framework for systematic measurement, data collection, and analysis
  - GOAL
    - Measurement objects can be products, processes and resources
  - QUESTION
    - Characterisation of the questions aligned with the objectives
  - METRIC
    - Measurements to answer the specified questions



V. Basili, R. Selby, and D. Hutchens. 1986. Experimentation in software engineering. *IEEE Trans. Softw. Eng.* 12, 7 (July 1986), 733-743.

Software Logging and Monitoring

- What is **logging**?
  - It is the practice of **recording relevant information** about a running system
  - **Logging statements**

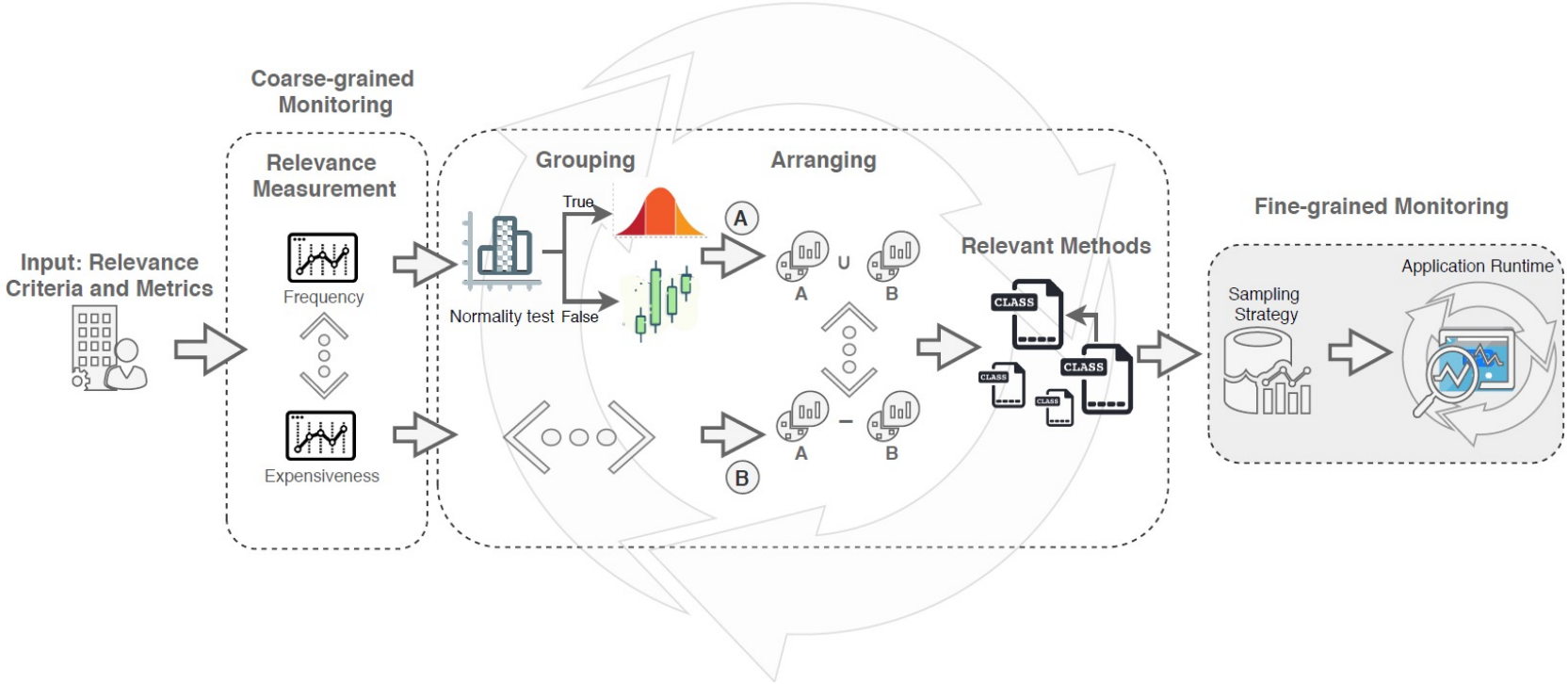
  - Be precise, concise and consistent in logging statements
  - Specify (in advance) and follow logging conventions

Software Logging and Monitoring

Goal Group		Efficiency	Maintainability	Reliability	Security	Testability
Goal Examples		performance, energy saving, caching, improving resource consumption	bug finding, understanding, reuse, documentation, troubleshooting	health checking, fault tolerance, disaster recovery, adaptation, configuration fix	anomaly detection, data protection, malicious attack detection	testing (generation, validation, selection), reporting, verification
Criterion	Frequency	Number of occurrences in a period	Number of references and dependencies	Inter-arrival times	Changes in occurrence history	Number of occurrences in test case
	Maintainability	Number of operations involved	Static source code metrics	—	Contextual information of objects/-classes	Fail test coverage
	Expensiveness	Execution time	Source code locations of expensive methods	CPU and heap utilization, processing times	Transaction duration	Depth of call stack
	Changeability	Number of repeated computations	Similarity between call graphs	Number of operations with cached results	Changes in contextual information	—
	Error-proneness	Number of failures of a component	Number of handled exceptions	Number of failures perceived by users	Increase of failures in a specific component	Number of failure assertions or exception-throwing statements
	Usage pattern	Changes in user navigational activity	—	Number of active users and idle/active intervals	Variations in the request payload for same operations	—
	State variation	I/O consumption per operation	Changes in the system state	Number of write operations performed	—	—
	Concurrency	Number of active users and threads	Number of references and dependencies	Number of race conditions	—	Number of locks per test case
	Latency	Processing and bandwidth consumption	—	Throughput	—	—

Software Logging and Monitoring

- Tigris: a two-phase framework for software tracing



MERTZ, J. ; NUNES, I. . Tigris: a DSL and Framework for Monitoring Software Systems at Runtime. The Journal of Systems & Software, 2021.

Summary

- Make an effective use of object orientation
- Have an architectural model
  - Software organisation and rules
- Adopt code review
- Manage technical debt
  - Make payments
- Use metrics and static analysis tools
- Collect runtime data
  - Consistent use of software logging
  - Know what data to collect with low performance impact

Challenges

- Quality
  - Conformance with requirements
    - Organisational requirements
    - Project requirements
      - Functional
      - Non-functional

Performance Bugs

Security Bugs

Challenges

- Performance Bugs
  - Finding and fixing performance issues
    - Search for equal or similar objects
    - Overuse of temporary structures
    - Containers used too little or too much
    - Data unnecessarily copied
    - Etc.
  - A software engineer's responsibility!

Cloud computing: high costs
The end of Moore's Law

Charles E. Leiserson. 2018. The Resurgence of Software Performance Engineering. SPAA '18. DOI:https://doi.org/10.1145/3210377.3210378

Challenges

- Performance Bugs
  - Use of application-level caching

Podcast: Fronteiras em Engenharia de Software: https://anchor.fm/fronteirases Episode #8

```
public class C1() {
  public Object process() {
  //creating the cache component
  Cache cache = Cache.getInstance();

  //looking up for cache content (steps in b)
  Object content = cache.get("c1:c2-computation");
  if (content == null){
    //cache miss (steps in c)
    content = C2.compute();

    //caching the content for future requests
    cache.set("c1:c2-computation", content);
  }

  //doing some business logic...
  return content;
  }
}
```

MERTZ, J. ; NUNES, I. . Understanding Application-Level Caching in Web Applications: A Comprehensive Introduction and Survey of State-of-the-Art Approaches. ACM COMPUTING SURVEYS, v. 50, p. 1-34, 2017.

Challenges

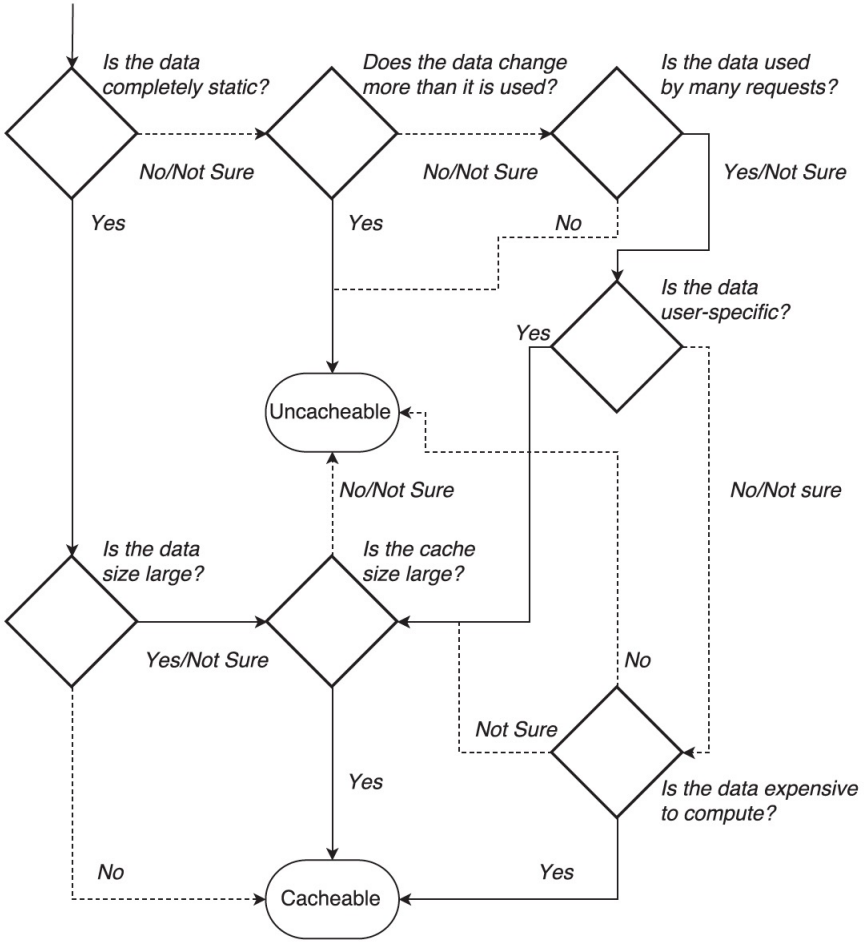
- Performance Bugs
  - Use of application-level caching

```
public class C1() {
  public Object process() {
  //creating the cache component
  Cache cache = Cache.getInstance();

  //looking up for cache content (steps in b)
  Object content = cache.get("c1:c2-computation");
  if (content == null){
    //cache miss (steps in c)
    content = C2.compute();

    //caching the content for future requests
    cache.set("c1:c2-computation", content);
  }

  //doing some business logic...
  return content;
  }
}
```



MERTZ, J. ; NUNES, I. . Understanding Application-Level Caching in Web Applications: A Comprehensive Introduction and Survey of State-of-the-Art Approaches. ACM COMPUTING SURVEYS, v. 50, p. 1-34, 2017.

Challenges

- Security Bugs

ID: UC_0001
Nome: Fazer login
Descrição: Usuário solicita fazer login no sistema, fornecendo usuário e senha, que são validados. Caso sejam válidos, usuário é logado no sistema. Caso contrário, o acesso é bloqueado. Mecanismos de segurança são adotados para impedir o acesso de usuários maliciosos.
Pré-condições: nenhum usuário está logado no sistema.
Pós-condições: usuário está registrado como logado no sistema.
Fluxo Básico
1. Este caso inicia quando o usuário seleciona a opção para fazer login no sistema. 2. O sistema solicita ao usuário o seu nome de usuário e a senha. 3. O usuário fornece os dados solicitados. 4. O sistema valida as regras de negócio RN1, RN2, RN3 e RN6. 5. O sistema registra o usuário como usuário logado no sistema. 6. O sistema exibe a tela inicial do sistema. 7. O caso de uso termina.
Fluxos Alternativos
Fluxo Alternativo 1 – Alternativa ao passo 3 3a.1. O usuário seleciona a opção de lembrete de nome de usuário ou senha. 3a.2. O sistema solicita e-mail cadastrado no sistema. 3a.3. O usuário fornece e-mail. 3a.4. O sistema valida a regra de negócio RN5. 3a.5. O sistema envia e-mail com o nome do usuário e opção para redefinir a senha. 3a.6. Retorna ao passo 2. **Fluxo Alternativo 2 – Alternativa ao passo 3a.4.** 3b.1. O sistema verifica que a regra de negócio RN5 não foi satisfeita. 3b.2. O sistema emite o alerta "O e-mail fornecido não é um endereço de e-mail válido." 3b.3. Retorna ao passo 3a.2.

Fluxo Alternativo 2 – Alternativa ao passo 3a.4.
3b.1. O sistema verifica que a regra de negócio RN5 não foi satisfeita.
3b.2. O sistema emite o alerta "O e-mail fornecido não é um endereço de e-mail válido."
3b.3. Retorna ao passo 3a.2.
Fluxo Alternativo 3 – Alternativa ao passo 4
4a.1. O sistema verifica que a regra de negócio RN1 não foi satisfeita.
4a.2. O sistema emite o alerta "Nome de usuário e senha são obrigatórios."
4a.3. Retorna ao passo 2.
Fluxo Alternativo 4 – Alternativa ao passo 4
4b.1. O sistema verifica que a regra de negócio RN2 não foi satisfeita.
4b.2. O sistema emite o alerta "Nome de usuário ou senha incorretos."
4b.3. Retorna ao passo 2.
Fluxo Alternativo 5 – Alternativa ao passo 4
4c.1. O sistema verifica que a regra de negócio RN3 não foi satisfeita.
4c.2. O sistema contabiliza uma tentativa de login para o usuário com nome de usuário fornecido.
4c.3. O sistema valida a regra de negócio RN4.
4c.4. O sistema emite o alerta "Nome de usuário ou senha incorretos."
4c.5. Retorna ao passo 2.
Fluxo Alternativo 6 – Alternativa ao passo 4c.3.
4d.1. O sistema verifica que a regra de negócio RN4 não foi satisfeita.
4d.2. O sistema bloqueia o usuário.
4d.3. O sistema emite o alerta "Usuário bloqueado: contate a nossa equipe de suporte."
4d.4. Retorna ao passo 2.
Fluxo Alternativo 7 – Alternativa ao passo 4.
4e.1. O sistema verifica que a regra de negócio RN6 não foi satisfeita.
4e.2. O sistema emite o alerta "Usuário bloqueado: contate a nossa equipe de suporte."
4e.3. Retorna ao passo 2.

Regras de Negócio

RN1. Um nome de usuário e uma senha foram fornecidos.
RN2. O nome de usuário fornecido corresponde a um nome de usuário cadastrado no sistema.
RN3. O hash da senha fornecida corresponde ao hash da senha cadastrada no sistema associada ao nome de usuário fornecido.
RN4. Podem ser feitas no máximo 3 tentativas de login de um usuário com senha incorreta.
RN5. O string fornecido é composto por mais de um caractere e possui @ no meio do string, e termina com ".com".
RN6. O usuário não se encontra bloqueado.

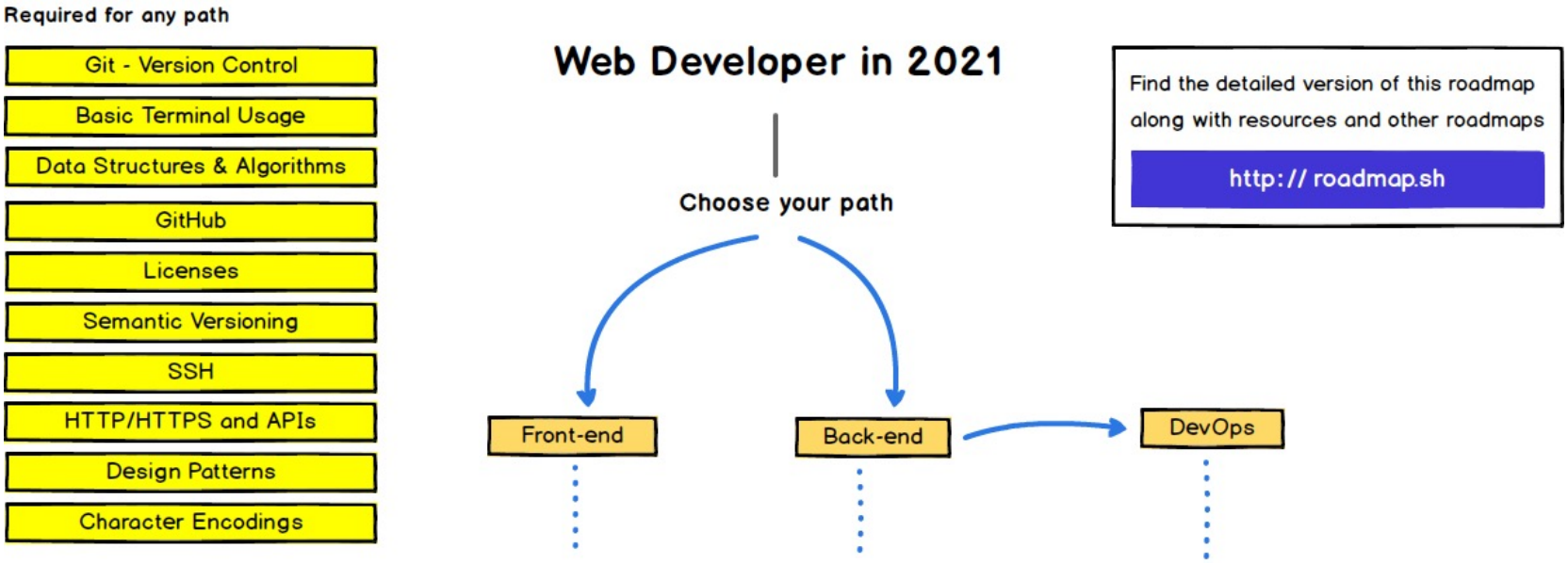
Requisitos Não-funcionais

Hashing da Senha: o algoritmo de hashing da senha deve ser SHA-3.
Tempo de resposta: o tempo de resposta de retorno quando as regras de negócio RN2 ou RN3 não forem válidas deve ser o mesmo.

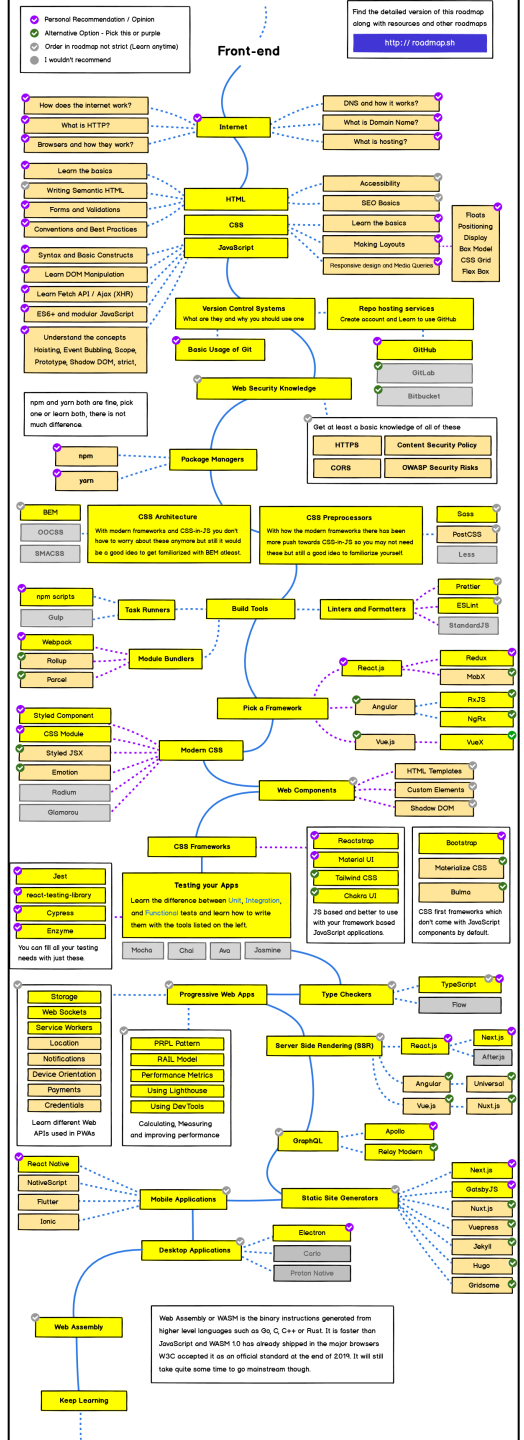
Challenges

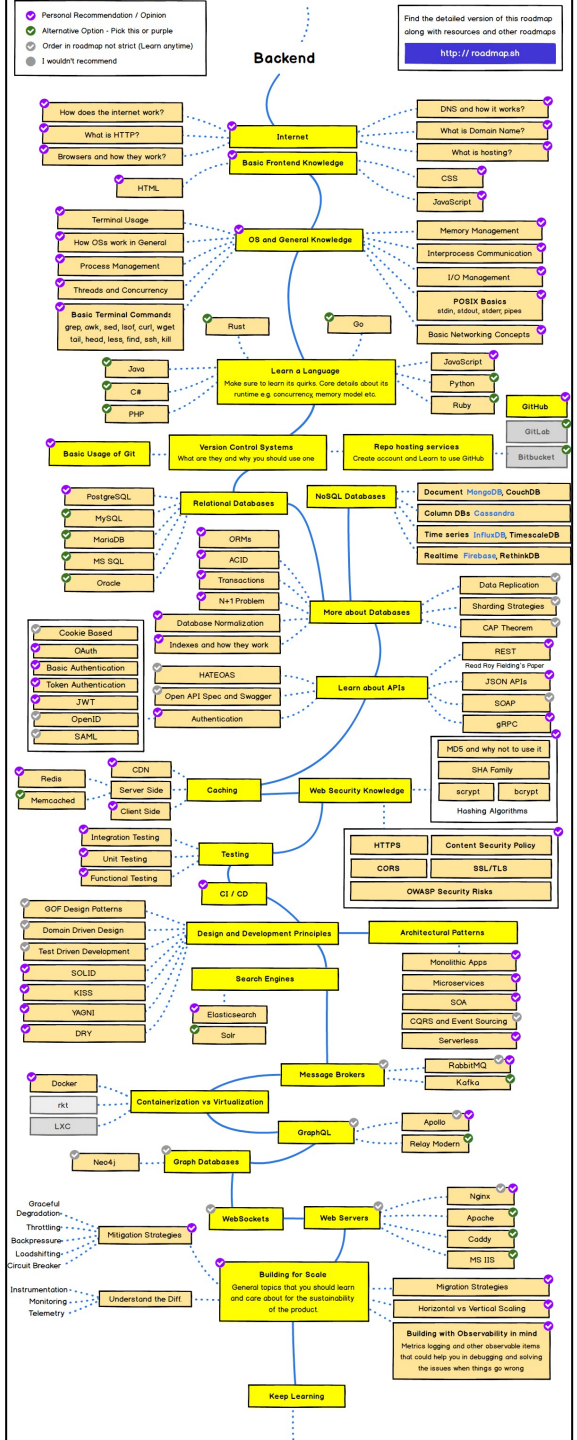

API Technologies
API Infrastructure
LAYER 7 TECHNOLOGIES
MuleSoft
MASHERY
RESTLET
apigee
3SCALE
API Middleware
OAuth.io
kloudless
singly
API Services
Runscope
import io
apiary
API Markets / Directories
mashape
programmableweb
Analytics and Testing
App Testing / Automation Technologies
Koality
CRITTERCISM
appurify
SAUCE LABS
APPLAUSE
Better data. Better apps.
Mobile App Analytics
mobile app tracking
UPSIGHT
FLURRY
swrve
Kontagent
Web App Analytics Technologies
Keen IO
mixpanel
App Performance Management
APPDYNAMICS
New Relic.
AppNeta
Coding Tools
Version Control
CVS
mercurial
SUBVERSION
git
Code Collaboration
Cloud9 IDE
Koding
github
Sourcegraph
runnable
VAGRANT
Open Source Code Frameworks
zend
RAILS
prototype
Meteor
django
Coding Environment
Atlassian
Dw
eclipse
JetBRAINS
NetBeans
Bowery
Open Source Coding Languages
Ruby
php
python
node
Continuous Integration
Jenkins
drone.io
circleci
Operating System Dev
Commercial Desktop OS
CANONICAL
redhat
Open Source Desktop OS
ubuntu
Linux
Mobile Device Dev
Windows Phone | Dev
BlackBerry Developer
iOS Dev Center
Developers
Data Dev Technologies
Big Data Dev Platforms
CONTINUITY
hp VERTICA
ORCHESTRATE
Cloudant
splunk>
Firebase
infochimps
NoSQL Technologies
Couchbase
mongoDB
AEROSPIKE
cassandra
riak
Hadoop Dev Technologies
wibidata
Hortonworks
cloudera
NewSQL Technologies
FOUNDATIONDB
memsql
NUODB
Clustrix
splice MACHINE
Graph Database Technologies
neotechnology
Objectivity
SQL Technologies
MySQL
PERCONA
ORACLE
Cloud Tools
Cloud Hosting
rackspace
the open cloud company
DreamHost
amazon web services S3
greenqloud
Processing-as-a-Service
amazon web services EC2
PubNub
REALTIME XRTML
Iron.io
User Cloud Services
Stormpath
DevOps Technologies
Server Management
CoreOS
linode.com
CFEngine
Ops Management
CHEF
catchpoint
radware
Apica
DATADOG
pingdom
Dyn
Content Delivery
fastly
CLOUDFLARE
edgecast
Log management
loggly
logentries
sumologic
Configuration Management
SALTSTACK
puppet labs
CHEF
Dev Platforms
Web App Platforms
heroku
Engine Yard
docker
elasticbox
Mobile App Platforms
built.io
Kii
famo.us
appcelerator
Parse
Sencha
kinvey
PhoneGap
Commercial Language Platform / Libraries
REVOLUTION ANALYTICS
Joyent
Typesafe
OPENSHIFT ONLINE

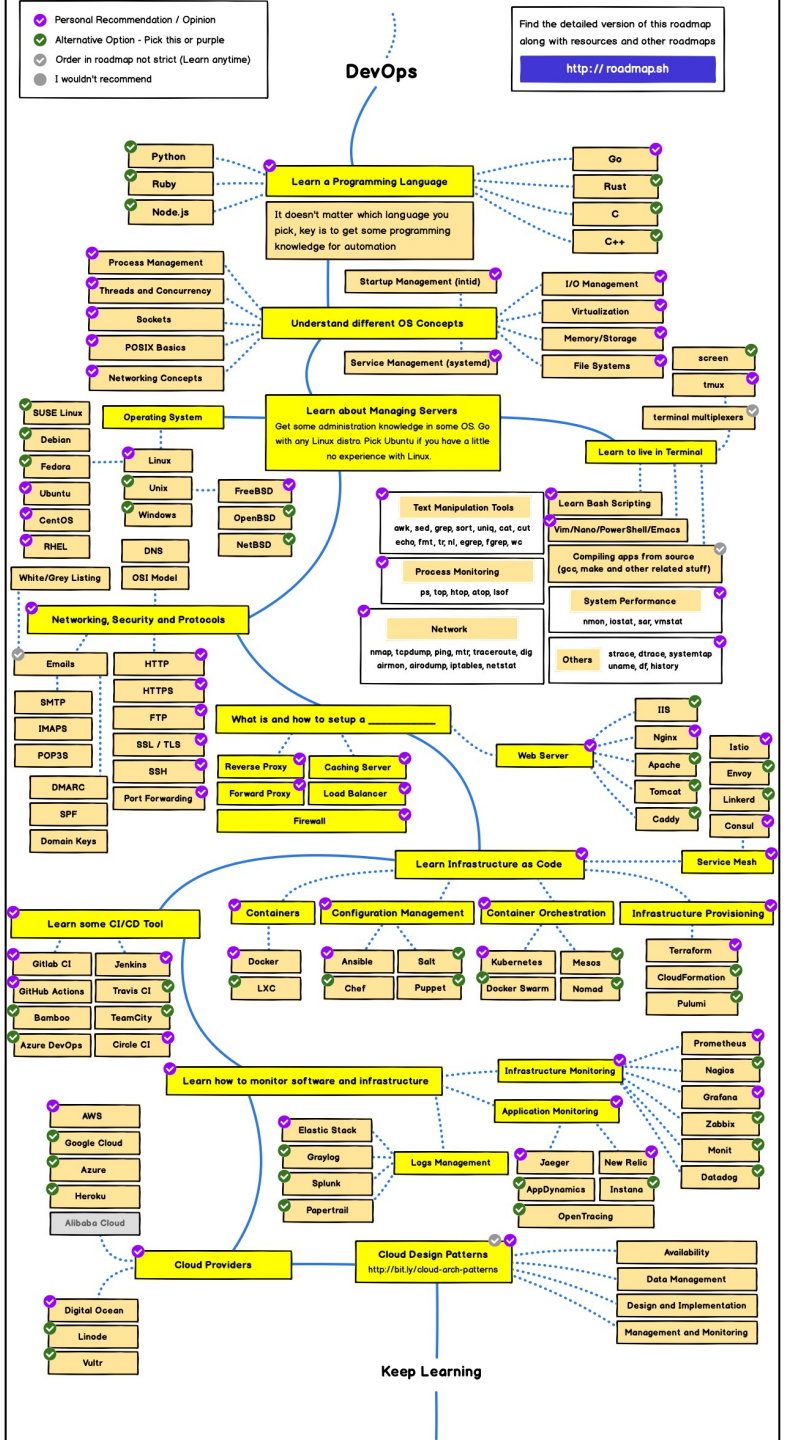
Challenges



https://github.com/kamranahmedse/developer-roadmap


Personal Recommendation / Opinion
Alternative Option - Pick this or purple
Order in roadmap not strict (Learn anytime)
I wouldn't recommend
Front-end
Find the detailed version of this roadmap along with resources and other roadmaps
http:// roadmap.sh
How does the internet work?
What is HTTP?
Browsers and how they work?
Internet
DNS and how it works?
What is Domain Name?
What is hosting?
Learn the basics
Writing Semantic HTML
Forms and Validations
Conventions and Best Practices
HTML
Accessibility
SEO Basics
Learn the basics
Making Layouts
Responsive design and Media Queries
CSS
Floats
Positioning
Display
Box Model
CSS Grid
Flex Box
JavaScript
Syntax and Basic Constructs
Learn DOM Manipulation
Learn Fetch API / Ajax (XHR)
ES6+ and modular JavaScript
Understand the concepts
Hoisting, Event Bubbling, Scope, Prototype, Shadow DOM, strict,
Version Control Systems
What are they and why you should use one
Repo hosting services
Create account and Learn to use GitHub
Basic Usage of Git
GitHub
GitLab
Bitbucket
Web Security Knowledge
Get at least a basic knowledge of all of these
HTTPS
Content Security Policy
CORS
OWASP Security Risks
npm and yarn both are fine, pick one or learn both, there is not much difference.
npm
yarn
Package Managers
BEM
OOCSS
SMACSS
CSS Architecture
With modern frameworks and CSS-in-JS you don't have to worry about these anymore but still it would be a good idea to get familiarized with BEM atleast.
CSS Preprocessors
With how the modern frameworks there has been more push towards CSS-in-JS so you may not need these but still a good idea to familiarize yourself.
Sass
PostCSS
Less
npm scripts
Gulp
Task Runners
Build Tools
Linters and Formatters
Prettier
ESLint
StandardJS
Webpack
Rollup
Parcel
Module Bundlers
React.js
Redux
MobX
Pick a Framework
Angular
RxJS
NgRx
Vue.js
VueX
Styled Component
CSS Module
Styled JSX
Emotion
Radium
Glamorou
Modern CSS
Web Components
HTML Templates
Custom Elements
Shadow DOM
CSS Frameworks
Reactstrap
Material UI
Tailwind CSS
Chakra UI
JS based and better to use with your framework based JavaScript applications.
Bootstrap
Materialize CSS
Bulma
CSS first frameworks which don't come with JavaScript components by default.
Jest
react-testing-library
Cypress
Enzyme
You can fill all your testing needs with just these.
Testing your Apps
Learn the difference between Unit, Integration, and Functional tests and learn how to write them with the tools listed on the left.
Mocha
Chai
Ava
Jasmine
Storage
Web Sockets
Service Workers
Location
Notifications
Device Orientation
Payments
Credentials
Learn different Web APIs used in PWAs
Progressive Web Apps
Type Checkers
TypeScript
Flow
PRPL Pattern
RAIL Model
Performance Metrics
Using Lighthouse
Using DevTools
Calculating, Measuring and improving performance
Server Side Rendering (SSR)
React.js
Next.js
After.js
Angular
Universal
Vue.js
Nuxt.js
GraphQL
Apollo
Relay Modern
React Native
NativeScript
Flutter
Ionic
Mobile Applications
Static Site Generators
Next.js
GatsbyJS
Nuxt.js
Vuepress
Jekyll
Hugo
Gridsome
Electron
Carlo
Proton Native
Desktop Applications
Web Assembly
Web Assembly or WASM is the binary instructions generated from higher level languages such as Go, C, C++ or Rust. It is faster than JavaScript and WASM 1.0 has already shipped in the major browsers W3C accepted it as an official standard at the end of 2019. It will still take quite some time to go mainstream though.
Keep Learning


Personal Recommendation / Opinion
Alternative Option - Pick this or purple
Order in roadmap not strict (Learn anytime)
I wouldn't recommend
Backend
Find the detailed version of this roadmap along with resources and other roadmaps
http:// roadmap.sh
How does the internet work?
What is HTTP?
Browsers and how they work?
Internet
DNS and how it works?
What is Domain Name?
What is hosting?
Basic Frontend Knowledge
HTML
CSS
JavaScript
Terminal Usage
How OSs work in General
Process Management
Threads and Concurrency
Basic Terminal Commands
grep, awk, sed, lsof, curl, wget
tail, head, less, find, ssh, kill
OS and General Knowledge
Memory Management
Interprocess Communication
I/O Management
POSIX Basics
stdin, stdout, stderr, pipes
Basic Networking Concepts
Rust
Go
Java
C#
PHP
Learn a Language
Make sure to learn its quirks. Core details about its runtime e.g. concurrency, memory model etc.
JavaScript
Python
Ruby
GitHub
GitLab
Bitbucket
Basic Usage of Git
Version Control Systems
What are they and why you should use one
Repo hosting services
Create account and Learn to use GitHub
PostgreSQL
MySQL
MariaDB
MS SQL
Oracle
Relational Databases
NoSQL Databases
Document MongoDB, CouchDB
Column DBs Cassandra
Time series InfluxDB, TimescaleDB
Realtime Firebase, RethinkDB
ORMs
ACID
Transactions
N+1 Problem
Database Normalization
Indexes and how they work
More about Databases
Data Replication
Sharding Strategies
CAP Theorem
Cookie Based
OAuth
Basic Authentication
Token Authentication
JWT
OpenID
SAML
HATEOAS
Open API Spec and Swagger
Authentication
Learn about APIs
REST
Read Roy Fielding's Paper
JSON APIs
SOAP
gRPC
Redis
Memcached
CDN
Server Side
Client Side
Caching
Web Security Knowledge
MD5 and why not to use it
SHA Family
scrypt
bcrypt
Hashing Algorithms
Integration Testing
Unit Testing
Functional Testing
Testing
CI / CD
HTTPS
Content Security Policy
CORS
SSL/TLS
OWASP Security Risks
GOF Design Patterns
Domain Driven Design
Test Driven Development
SOLID
KISS
YAGNI
DRY
Design and Development Principles
Architectural Patterns
Search Engines
Elasticsearch
Solr
Monolithic Apps
Microservices
SOA
CQRS and Event Sourcing
Serverless
Docker
rkt
LXC
Containerization vs Virtualization
Message Brokers
RabbitMQ
Kafka
GraphQL
Apollo
Relay Modern
Neo4j
Graph Databases
Graceful Degradation
Throttling
Backpressure
Loadshifting
Circuit Breaker
Mitigation Strategies
WebSockets
Web Servers
Nginx
Apache
Caddy
MS IIS
Instrumentation
Monitoring
Telemetry
Understand the Diff
Building for Scale
General topics that you should learn and care about for the sustainability of the product.
Migration Strategies
Horizontal vs Vertical Scaling
Building with Observability in mind
Metrics logging and other observable items that could help you in debugging and solving the issues when things go wrong
Keep Learning


Personal Recommendation / Opinion
Alternative Option - Pick this or purple
Order in roadmap not strict (Learn anytime)
I wouldn't recommend
DevOps
Find the detailed version of this roadmap along with resources and other roadmaps
http:// roadmap.sh
Python
Ruby
Node.js
Learn a Programming Language
It doesn't matter which language you pick, key is to get some programming knowledge for automation
Go
Rust
C
C++
Process Management
Threads and Concurrency
Sockets
POSIX Basics
Networking Concepts
Understand different OS Concepts
Startup Management (intid)
Service Management (systemd)
I/O Management
Virtualization
Memory/Storage
File Systems
screen
tmux
terminal multiplexers
SUSE Linux
Debian
Fedora
Ubuntu
CentOS
RHEL
Operating System
Linux
Unix
Windows
DNS
OSI Model
FreeBSD
OpenBSD
NetBSD
White/Grey Listing
Learn about Managing Servers
Get some administration knowledge in some OS. Go with any Linux distro. Pick Ubuntu if you have a little no experience with Linux.
Learn to live in Terminal
Text Manipulation Tools
awk, sed, grep, sort, uniq, cat, cut
echo, fmt, tr, nl, egrep, fgrep, wc
Learn Bash Scripting
Vim/Nano/PowerShell/Emacs
Process Monitoring
ps, top, htop, atop, lsof
Compiling apps from source (gcc, make and other related stuff)
System Performance
nmon, iostat, sar, vmstat
Network
nmap, tcpdump, ping, mtr, traceroute, dig
airmon, airodump, iptables, netstat
Others
strace, dtrace, systemtap
uname, df, history
Networking, Security and Protocols
Emails
SMTP
IMAPS
POP3S
DMARC
SPF
Domain Keys
HTTP
HTTPS
FTP
SSL / TLS
SSH
Port Forwarding
What is and how to setup a _____
Reverse Proxy
Caching Server
Forward Proxy
Load Balancer
Firewall
Web Server
IIS
Nginx
Apache
Tomcat
Caddy
Istio
Envoy
Linkerd
Consul
Learn Infrastructure as Code
Service Mesh
Learn some CI/CD Tool
Containers
Configuration Management
Container Orchestration
Infrastructure Provisioning
Gitlab CI
GitHub Actions
Bamboo
Azure DevOps
Jenkins
Travis CI
TeamCity
Circle CI
Docker
LXC
Ansible
Chef
Salt
Puppet
Kubernetes
Docker Swarm
Mesos
Nomad
Terraform
CloudFormation
Pulumi
Learn how to monitor software and infrastructure
Infrastructure Monitoring
Application Monitoring
Logs Management
Elastic Stack
Graylog
Splunk
Papertrail
Jaeger
New Relic
AppDynamics
Instana
OpenTracing
Prometheus
Nagios
Grafana
Zabbix
Monit
Datadog
AWS
Google Cloud
Azure
Heroku
Alibaba Cloud
Cloud Providers
Digital Ocean
Linode
Vultr
Cloud Design Patterns
http://bit.ly/cloud-arch-patterns
Availability
Data Management
Design and Implementation
Management and Monitoring
Keep Learning

Challenges

- Heterogeneous world of technologies and programming languages
  - OO and functional languages
  - Compiled and interpreted languages
  - Strongly and dynamically typed languages
  - General purpose and domain-specific languages

Our central finding is that both static type systems find an important percentage of public bugs: both Flow 0.30 and TypeScript 2.0 successfully detect 15%!

Zheng Gao, Christian Bird, and Earl T. Barr. 2017. To type or not to type: quantifying detectable bugs in JavaScript. In Proceedings of the 39th International Conference on Software Engineering (ICSE '17). IEEE Press, Piscataway, NJ, USA, 758-769. DOI: https://doi.org/10.1109/ICSE.2017.75

Challenges

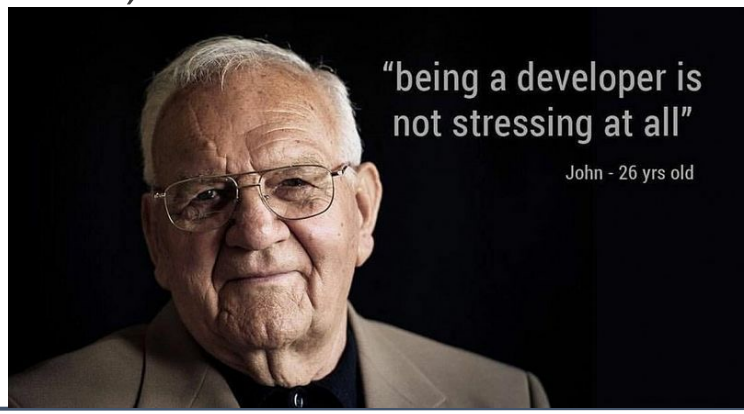
All project decisions must be well informed! Too bad that TECHNICAL decisions often become BUSINESS decisions.

Final Considerations

- Most of the content of this talk are covered in undergraduate courses
  - Problem: theory vs. practice
  - Role of internship (at least, in Brazil)?

Programmers vs. Software Engineers

- To give importance to code maintainability and legibility
  - Cost reduction (less bugs, easier evolution)
  - Happy programmers



Final Considerations

- Most of the content of this talk are covered in undergraduate courses
  - Problem: theory vs. practice
  - Role of internship (at least, in Brazil)?

Programmers vs. Software Engineers

- To give importance to code maintainability and legibility
  - Cost reduction (less bugs, easier evolution)
  - Happy programmers

Despite all advances, why does *go horse* still exist?

Final Considerations

- How to help us?
  - Access to projects and developers for research
    - Project mining (with access to the source code or issue trackers)
    - Surveys with developers
    - It is possible to sign NDAs (Non-Disclosure Agreement)
      - Publicatins may or may not include acknowledgements
      - Annonymised data (after the company's approval)

Thanks!

- Prof. Ingrid Nunes (UFRGS)
  - Homepage
    http://inf.ufrgs.br/~ingridnunes/
  - Twitter
    https://twitter.com/ingridnunesIN
  - Facebook
    https://www.facebook.com/ingridnunesIN
  - LinkedIn
    https://www.linkedin.com/in/ingrid-nunes/

Why does *go horse* still exist?