



# Code Quality: *Raiz vs. Nutella*



Prof. Ingrid Nunes

Universidade Federal do Rio Grande do Sul (UFRGS)

Promob Dev Day – May 28, 2021



Stack Overflow 

@StackOverflow



. @rla4, the tech lead on Stack Overflow for Teams, explains why we **ignored** several **best practices** when building Stack Overflow's public site 12 years ago, and how we're **modernizing our codebase** to be approachable and powerful today.



The Stack Overflow Podcast

We chat with Roberta Arcoverde, the tech lead on Stack Overflow for Teams. She explains why we ignored several ...

[the-stack-overflow-podcast.simplecast.com](https://the-stack-overflow-podcast.simplecast.com)

5:40 PM · Mar 30, 2021 · Hootsuite Inc.

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

**Why?**



**Software development is still  
a highly human-based activity.**

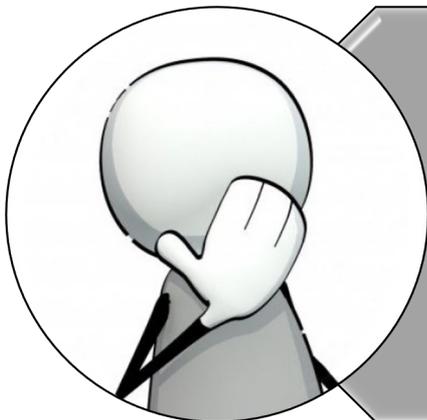




# Two Messages



Process and practices are helpful for the ability of **repetition** and **likelihood** of **success**.



We must make sure we invest **time to think** through all artefacts that we produce.

**FIRST FIRST**  
**THINGS FIRST**

# Let's talk about requirements



<b>ID:</b> UC_0001
<b>Nome:</b> Fazer login
<b>Descrição:</b> Usuário solicita fazer login no sistema que são validados.
<b>Pré-condições:</b> nenhum usuário está logado no sistema.
<b>Pós-condições:</b> usuário está registrado como logado no sistema.
<b>Fluxo Básico</b>
<ol style="list-style-type: none"><li>1. Este caso inicia quando o usuário seleciona a opção para fazer login no sistema.</li><li>2. O sistema solicita ao usuário o seu nome de usuário e a senha.</li><li>3. O usuário fornece os dados solicitados.</li><li>4. O sistema valida os dados fornecidos.</li><li>5. O sistema registra o usuário como usuário logado no sistema.</li><li>6. O sistema exibe a tela inicial do sistema.</li><li>7. O caso de uso termina.</li></ol>
<b>Fluxos Alternativos</b>
<b>Fluxo Alternativo 1 – Alternativa ao passo 4</b>
<ol style="list-style-type: none"><li>4a.1. O sistema verifica que os dados estão incorreto.</li><li>4a.2. O sistema notifica ao usuário.</li><li>4a.3. Retorna ao passo 2.</li></ol>

# Let's talk about requirements



As a registered user <role>  
.....

I want ..... to login <goal>,  
.....

So that I can access <benefit>  
.....  
all system features

Acceptance criteria:  
.....  
.....  
?

**The point is  
not the  
notation but  
the content!**

# Let's talk about requirements



<b>ID:</b> UC_0001
<b>Nome:</b> Fazer login
<b>Descrição:</b> 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.
<b>Pré-condições:</b> nenhum usuário está logado no sistema.
<b>Pós-condições:</b> usuário está registrado como logado no sistema.
<b>Fluxo Básico</b>
<ol style="list-style-type: none"><li>1. Este caso inicia quando o usuário seleciona a opção para fazer login no sistema.</li><li>2. O sistema solicita ao usuário o seu nome de usuário e a senha.</li><li>3. O usuário fornece os dados solicitados.</li><li>4. O sistema valida as regras de negócio RN1, RN2, RN3 e RN6.</li><li>5. O sistema registra o usuário como usuário logado no sistema.</li><li>6. O sistema exibe a tela inicial do sistema.</li><li>7. O caso de uso termina.</li></ol>
<b>Fluxos Alternativos</b>
<b>Fluxo Alternativo 1 – Alternativa ao passo 3</b>
<ol style="list-style-type: none"><li>3a.1. O usuário seleciona a opção de lembrete de nome de usuário ou senha.</li><li>3a.2. O sistema solicita e-mail cadastrado no sistema.</li><li>3a.3. O usuário fornece e-mail.</li><li>3a.4. O sistema valida a regra de negócio RN5.</li><li>3a.5. O sistema envia e-mail com o nome do usuário e opção para redefinir a senha.</li><li>3a.6. Retorna ao passo 2.</li></ol>
<b>Fluxo Alternativo 2 – Alternativa ao passo 3a.4.</b>
<ol style="list-style-type: none"><li>3b.1. O sistema verifica que a regra de negócio RN5 não foi satisfeita.</li><li>3b.2. O sistema emite o alerta "O e-mail fornecido não é um endereço de e-mail válido."</li><li>3b.3. Retorna ao passo 3a.2.</li></ol>
<b>Fluxo Alternativo 3 – Alternativa ao passo 4</b>
<ol style="list-style-type: none"><li>4a.1. O sistema verifica que a regra de negócio RN1 não foi satisfeita.</li><li>4a.2. O sistema emite o alerta "Nome de usuário e senha são obrigatórios."</li><li>4a.3. Retorna ao passo 2.</li></ol>
<b>Fluxo Alternativo 4 – Alternativa ao passo 4</b>
<ol style="list-style-type: none"><li>4b.1. O sistema verifica que a regra de negócio RN2 não foi satisfeita.</li><li>4b.2. O sistema emite o alerta "Nome de usuário ou senha incorretos."</li><li>4b.3. Retorna ao passo 2.</li></ol>
<b>Fluxo Alternativo 5 – Alternativa ao passo 4</b>
<ol style="list-style-type: none"><li>4c.1. O sistema verifica que a regra de negócio RN3 não foi satisfeita.</li><li>4c.2. O sistema contabiliza uma tentativa de login para o usuário com nome de usuário fornecido.</li><li>4c.3. O sistema valida a regra de negócio RN4.</li><li>4c.4. O sistema emite o alerta "Nome de usuário ou senha incorretos."</li><li>4c.5. Retorna ao passo 2.</li></ol>
<b>Fluxo Alternativo 6 – Alternativa ao passo 4c.3.</b>
<ol style="list-style-type: none"><li>4d.1. O sistema verifica que a regra de negócio RN4 não foi satisfeita.</li><li>4d.2. O sistema bloqueia o usuário.</li><li>4d.3. O sistema emite o alerta "Usuário bloqueado: contate a nossa equipe de suporte."</li><li>4d.4. Retorna ao passo 2.</li></ol>
<b>Fluxo Alternativo 7 – Alternativa ao passo 4.</b>
<ol style="list-style-type: none"><li>4e.1. O sistema verifica que a regra de negócio RN6 não foi satisfeita.</li><li>4e.2. O sistema emite o alerta "Usuário bloqueado: contate a nossa equipe de suporte."</li><li>4e.3. Retorna ao passo 2.</li></ol>
<b>Regras de Negócio</b>
<p><b>RN1.</b> Um nome de usuário e uma senha foram fornecidos. <b>RN2.</b> O nome de usuário fornecido corresponde a um nome de usuário cadastrado no sistema. <b>RN3.</b> O hash da senha fornecida corresponde ao hash da senha cadastrada no sistema associada ao nome de usuário fornecido. <b>RN4.</b> Podem ser feitas no máximo 3 tentativas de login de um usuário com senha incorreta. <b>RN5.</b> O string fornecido é composto por mais de um caractere e possui @ no meio do string, e termina com ".com". <b>RN6.</b> O usuário não se encontra bloqueado.</p>
<b>Requisitos Não-funcionais</b>
<p><b>Hashing da Senha:</b> o algoritmo de hashing da senha deve ser SHA-3. <b>Tempo de resposta:</b> o tempo de resposta de retorno quando as regras de negócio RN2 ou RN3 não forem válidas deve ser o mesmo.</p>

# Let's talk about requirements



<b>ID:</b> UC_0001
<b>Nome:</b> Fazer login
<b>Descrição:</b> 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.
<b>Pré-condições:</b> nenhum usuário está logado no sistema.
<b>Pós-condições:</b> usuário está registrado como logado no sistema.
<b>Fluxo Básico</b>
<ol style="list-style-type: none"><li>1. Este caso inicia quando o usuário seleciona a opção para fazer login no sistema.</li><li>2. O sistema solicita ao usuário o seu nome de usuário e a senha.</li><li>3. O usuário fornece os dados solicitados.</li><li>4. O sistema valida as regras de negócio RN1, RN2, RN3 e RN6.</li><li>5. O sistema registra o usuário como usuário logado no sistema.</li><li>6. O sistema exibe a tela inicial do sistema.</li><li>7. O caso de uso termina.</li></ol>
<b>Fluxos Alternativos</b>
<b>Fluxo Alternativo 1 – Alternativa ao passo 3</b> <ol style="list-style-type: none"><li>3a.1. O usuário seleciona a opção de lembrete de nome de usuário ou senha.</li><li>3a.2. O sistema solicita e-mail cadastrado no sistema.</li><li>3a.3. O usuário fornece e-mail.</li><li>3a.4. O sistema valida a regra de negócio RN5.</li><li>3a.5. O sistema envia e-mail com o nome do usuário e opção para redefinir a senha.</li><li>3a.6. Retorna ao passo 2.</li></ol>

## 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 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 o passo seguinte do suporte.”





- 4c.1. O sistema verifica que a regra de negócio RN5 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

**RN5.** O string fornecido é composto por mais de um caractere 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 o mesmo.

**Tempo de resposta:** o tempo de resposta de retorno quando as regras de negócio forem válidas deve ser o mesmo.

In agile methods, all this information must be detailed in **acceptance criteria** and **test cases!**

**But... wasn't  
the talk about  
code quality?**

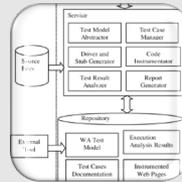


# Importance of (well-understood) requirements

- **Prevent unpredicted** challenges while designing and coding
- Enables a **proper design** of the solution
  - Rather than implementing bit by bit
- Prevent **reworking** the code
- Crucial for designing **test cases**

Software without well-specified requirements cannot be right or wrong, it can only surprise you.

# Key Points



Have an architectural model



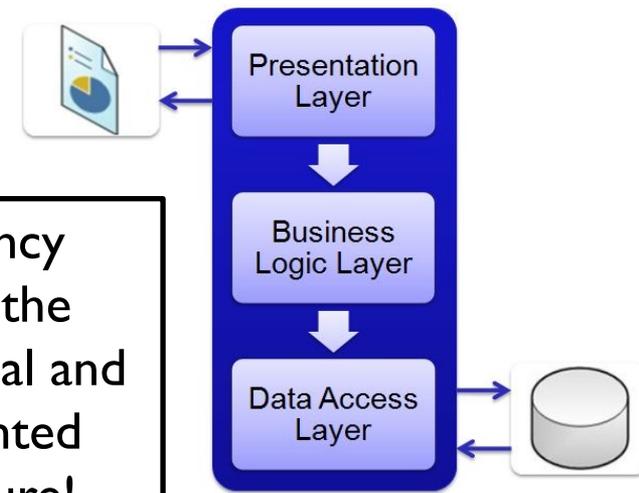
# Software Architecture

"The software architecture of a program or computing system is the **structure or structures of the system**, which comprise software elements, the **externally visible properties** of those elements, and the **relationships** among them."

- Bass et al. 2003

- Modules
  - Including design-time, test-time, and run-time hardware and software parts
- Externally visible properties
  - Modules with interfaces, hardware units, objects
- Relationships and constraints
  - Dependencies

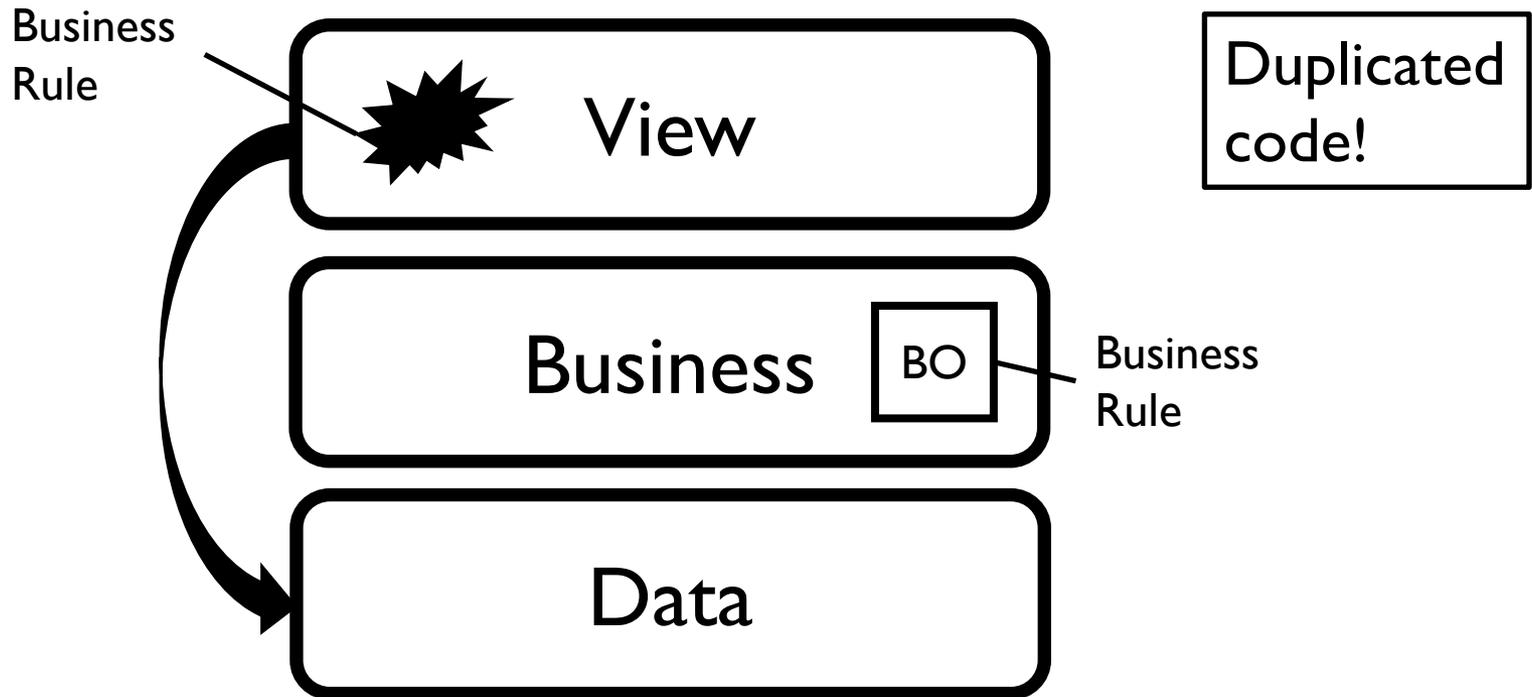
Consistency between the conceptual and implemented architecture!





# Software Architecture

- Fundamental for an organised software evolution





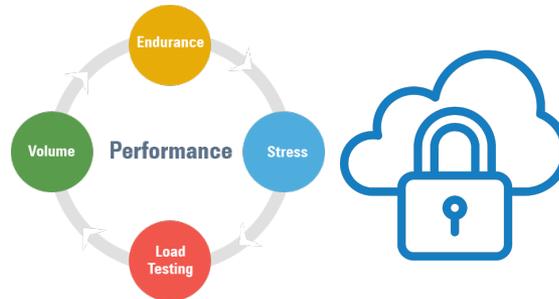
# Software Architecture

- Key design decisions
  - Adopted technologies



- Non-functional requirements

Scalability



# Software Architecture



Phil Calçado  
@pcalcado

Monoliths are fine if you are committed to them.

(Micro)services are fine if you are committed to them.

Microliths are what happens when an organization isn't brave enough to pick a lane. The worst of both worlds without the advantages of any of them.

7:49 PM · Mar 26, 2021 · TweetDeck

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



...

BRUNO CARTAXO  
**HIDEV** PODCAST

**a arquitetura do stack overflow é monolítica. é muito adequado a nossa história de escalabilidade. a gente não é e-commerce.**

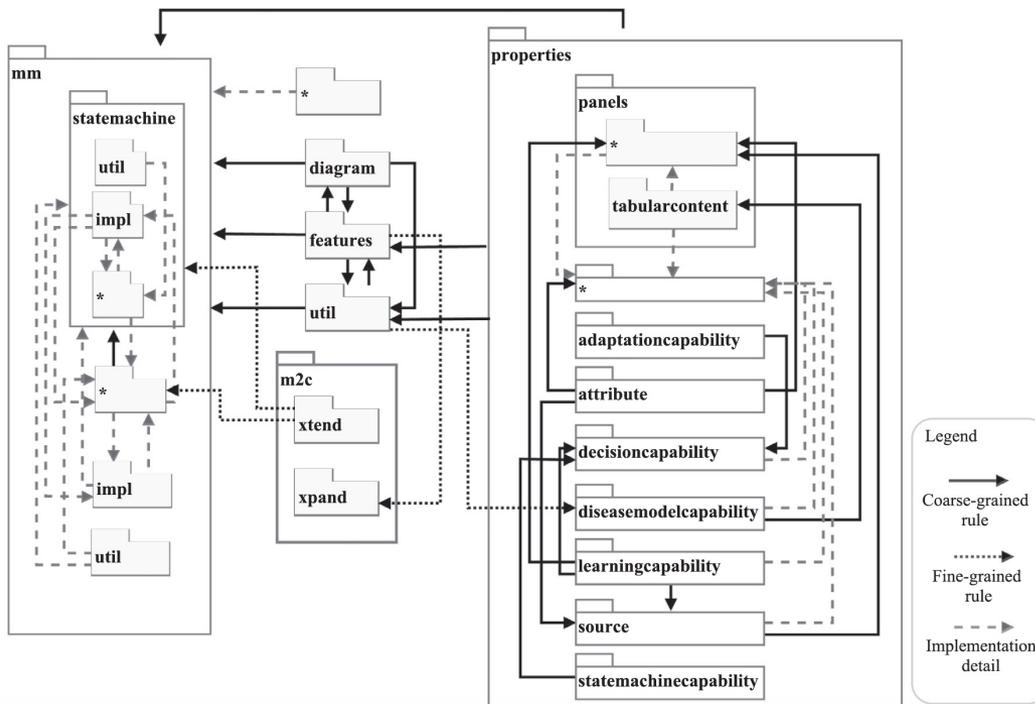
ROBERTA ARCOVERDE

284 views 0:12 / 2:09

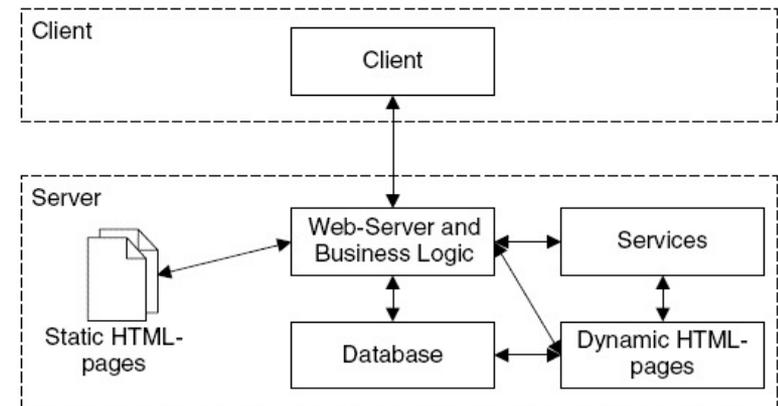
# Software Architecture



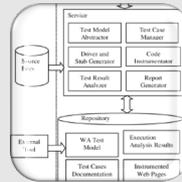
## Raiz



## Nutella



# Key Points



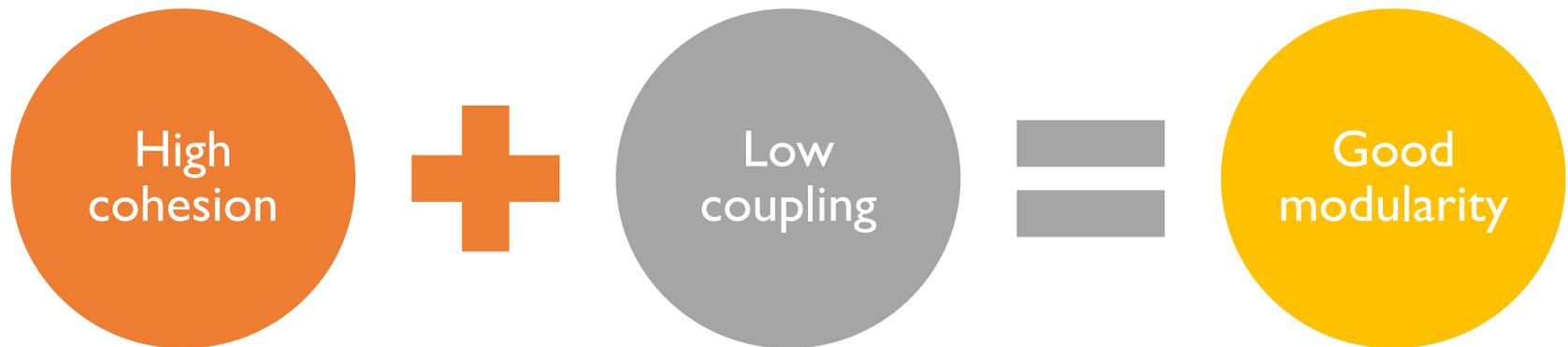
Have an architectural model



Do not neglect modularisation and separation of concerns



# Modularity and SoC





# Modularity and SoC

```
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?



# Modularity and SoC

```
class UserController {  
  
    createUser() {  
        // check data valid  
        new User()  
    }  
  
    resetPassword() {  
        // check token valid  
        // check pw valid  
        user.setPassword(newPWV)  
    }  
  
}
```

```
class User {  
  
    login  
    password  
  
    //getters  
  
    //setters  
  
}
```

Duplicated code!  
High coupling!



# Modularity and SoC

```
class UserController {
```

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

```
    resetPassword() {  
        // check token valid  
        checkUser()  
    }
```

```
}
```

```
class User {
```

```
    login  
    password
```

```
    //getters
```

```
    //setters
```

```
}
```

```
class UserValidator {
```

```
    checkUser() { ... }
```

```
}
```

**Procedural  
Paradigm ???**



# Modularity and SoC

- Effective use of object orientation

```
class UserController {  
  
    createUser() {  
        new User()  
    }  
  
    resetPassword() {  
        user.resetPassword(newPWV)  
    }  
  
}
```

```
class User {  
  
    login  
    password  
  
    User () {  
        ...  
        setPassword()  
    }  
    setPassword() {  
        //check pw valid  
    }  
    resetPassword() {  
        isTokenValid()  
        setPassword()  
    }  
    isTokenValid() { ... }  
}
```



# Modularity and SoC

- 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.
  - Single Responsibility Principles
  - Open-Closed Principle
  - Liskov Substitution Principle
  - Interface Segregation Principle
  - Dependency Inversion Principle



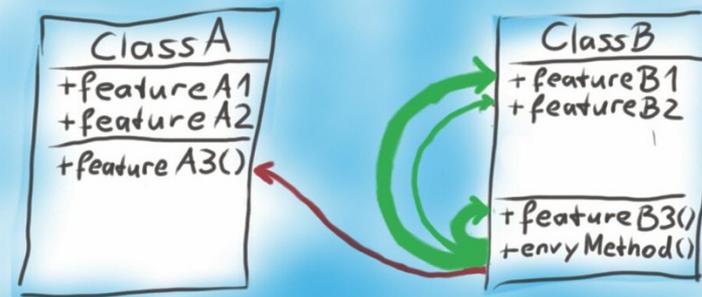
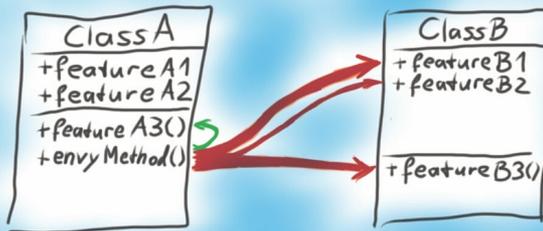
# Modularity and SoC

- GRASP Patterns

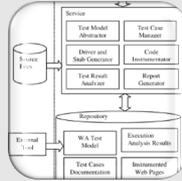
- E.g. Specialist
- Craig Larman. 1997. Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development.

- Code Smells / Refactoring Catalog

- E.g.
  - Feature Envy
  - Data Class
- Martin Fowler. 1999. Refactoring: Improving the Design of Existing Code.



# Key Points



Have an architectural model



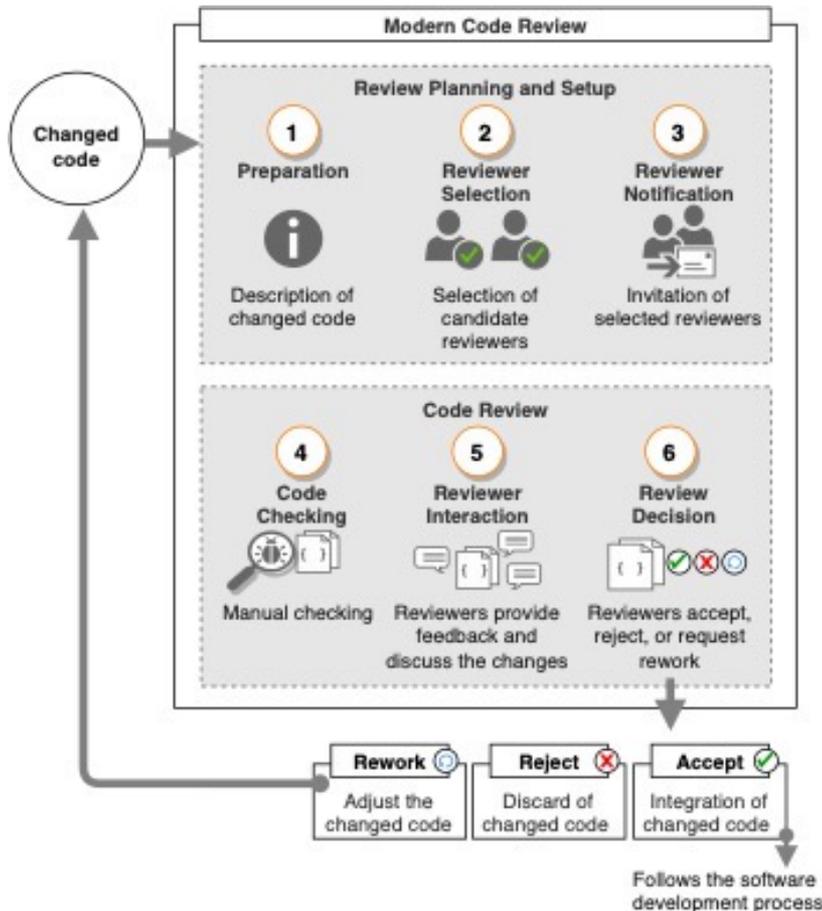
Do not neglect modularisation and separation of concerns



Adopt code review (and check the points above!)



# Code Review



- Effective verification technique
  - Code quality
  - Knowledge sharing
  - Code ownership



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



# Code Review

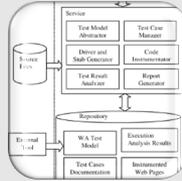
- Code readability and understandability  $\neq$  good modularity

“To do this, we use 444 human evaluations from 63 developers and we obtained a bold negative result: **none** of the **121** experimented metrics is able to **capture code understandability**, not even the ones assumed to assess quality attributes apparently related, such as code readability and complexity.”



S. Scalabrino, G. Bavota, C. Vendome, M. Linares-Vásquez, D. Poshyvanyk and R. Oliveto, "Automatically Assessing Code Understandability," in IEEE Transactions on Software Engineering, 2021

# Key Points



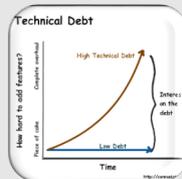
Have an architectural model



Do not neglect modularisation and separation of concerns

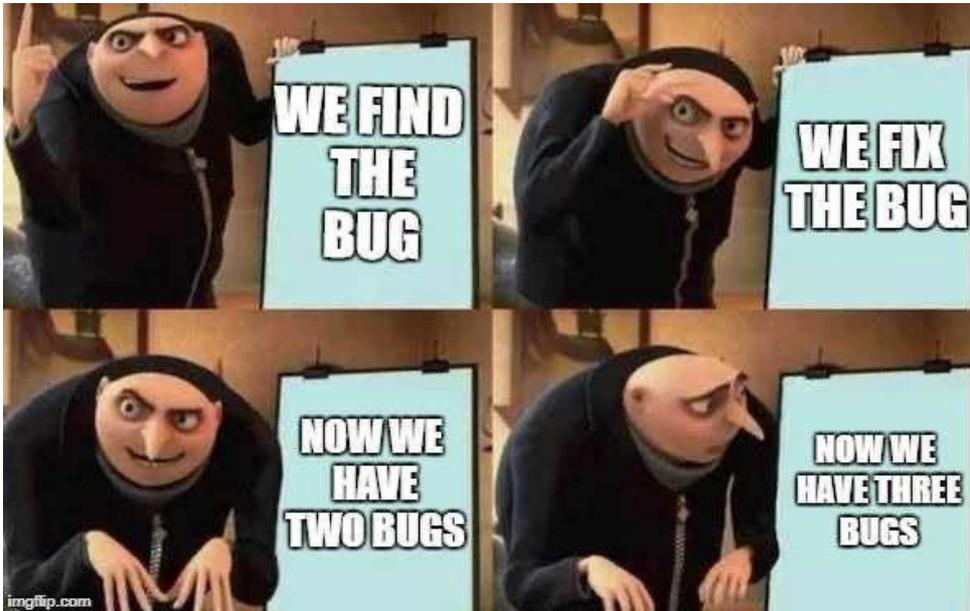


Adopt code review (and check the points above!)



Manage technical debt (and make payments!)

# Technical Debt



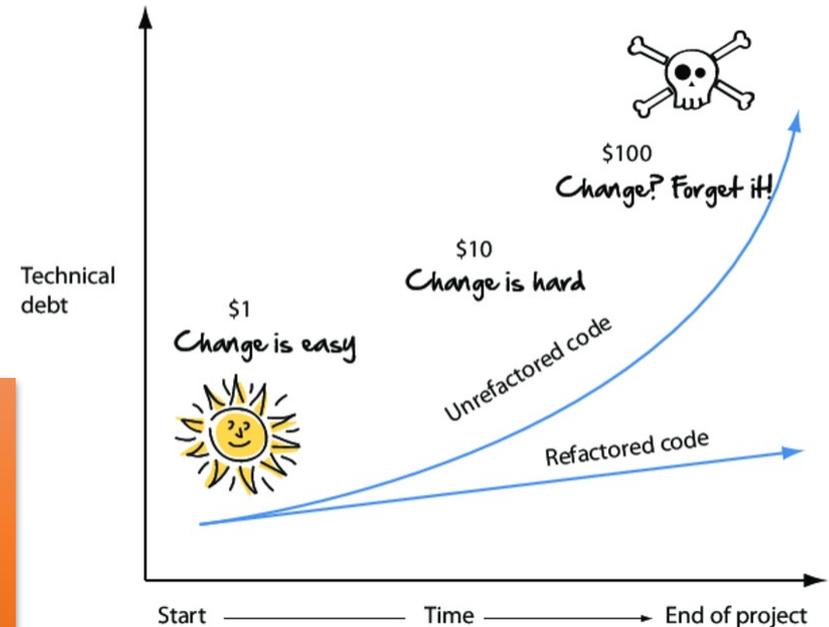
© www.SoftwareTestingHelp.com



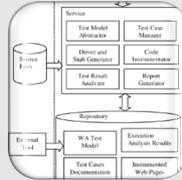
# Technical Debt

- 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**.



# Key Points



Have an architectural model



Do not neglect modularisation and separation of concerns



Adopt code review (and check the points above!)



Manage technical debt (and make payments!)



Use metrics and static analysis tools (automation!)

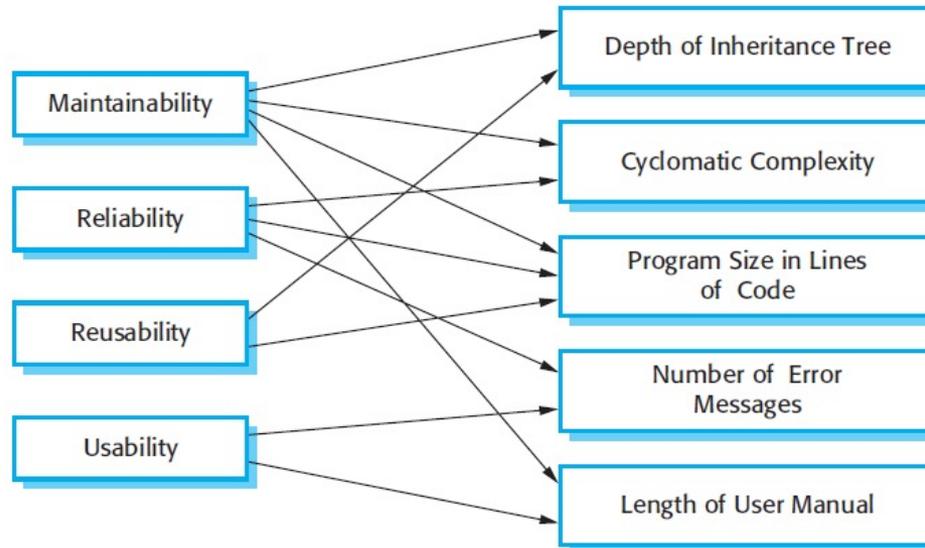


# Metrics and Static Analysis Tools

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

External Quality Attributes

Internal Attributes





# • 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**, e.g. dependencies, code smell detections
  - Identification of **anomalies**
  - Part of **automated reviewers**

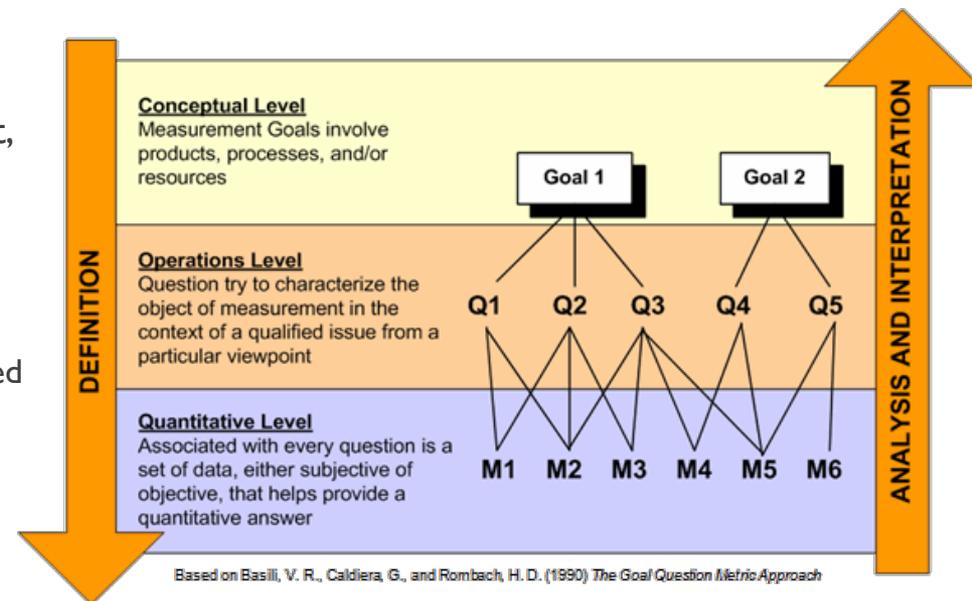
# Metrics and Static Analysis Tools



“You can’t manage what you can’t measure”,

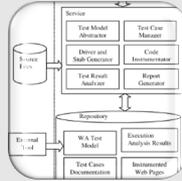
Tom DeMarco

- 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.

# Key Points



Have an architectural model



Do not neglect modularisation and separation of concerns



Adopt code review (and check the points above!)



Manage technical debt (and make payments!)



Use metrics and static analysis tools (automation!)



Software logging (with guidelines and standards!)





# Logging Best Practices: The 13 You Should Know

1. Don't Write Logs by Yourself (AKA Don't Reinvent the Wheel)
2. Log at the Proper Level
3. Employ the Proper Log Category
4. Write Meaningful Log Messages
5. Write Log Messages in English
6. Add Context to Your Log Messages
7. Log in Machine Parseable Format
8. But Make the Logs Human-Readable as Well
9. Don't Log Too Much or Too Little
10. Think of Your Audience
11. Don't Log for Troubleshooting Purposes Only
12. Avoid Vendor Lock-In
13. Don't Log Sensitive Information

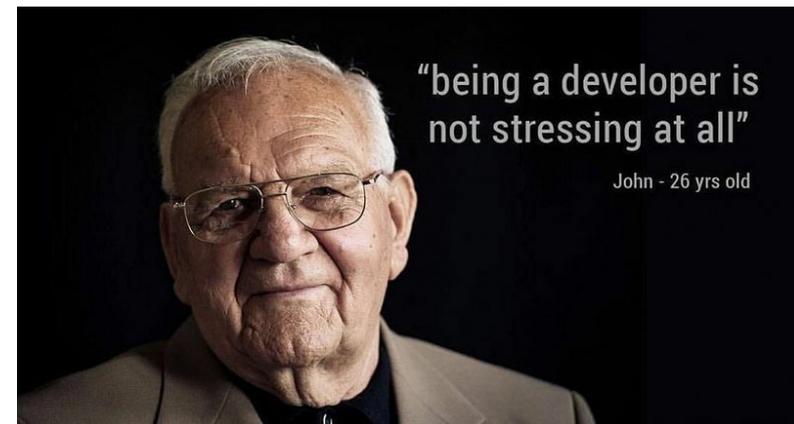


Dave McAllister. <https://www.scalyr.com/blog/the-10-commandments-of-logging/> October, 2019.



# Final Considerations

- Most of the content of this talk are covered in undergraduate courses
  - Problem: theory vs. practice
- To give importance to code maintainability and legibility
  - Cost reduction (less bugs, easier evolution)
  - Happy programmers 
  - How to convince management and those responsible for business decisions?





# Software Development Effort Estimation



# Mental Health of Professionals of Software Development



Do you think that working on software development is **stressful**? Do you want to know how jobs related to **SOFTWARE DEVELOPMENT** affect a person's **MOOD** and **MENTAL HEALTH**?

We would like to find this out but, for that, we need the help of people from the software industry. If you are one, happy or not with your job, please complete our survey! We also would REALLY appreciate it if you share our invitation with your peers so we can reach a high number of participants :)

Link: <https://forms.gle/NBTVuc5svbGfEBfz6>  
(estimated time: 20 min)



# Podcasts



Emílias - Armação em Bits

@Emílias\_UTFPR



Saiu o novo episódio do Emílias Podcast em que entrevistamos @ingridnunesIN, nova co-host do

@FronteirasES :

[open.spotify.com/episode/4gCd8c...](https://open.spotify.com/episode/4gCd8c...)

#MulheresPodcasters

#MulheresNaComputacao

Se gostar, dá RT!

cc @rla4

[Translate Tweet](#)



Ingrid Nunes: Professora da UFRGS - Em Bits (UTFPR Curitiba)

[open.spotify.com](https://open.spotify.com)



Revelando o Dark side: lições aprendidas em uma luta contra depressão - Profa. Ingrid Nunes (UFRGS)

1,606 views · Streamed live on 29 Jul 2020

[Like](#) 102 [Comment](#) 1 [Share](#) [Save](#) [More](#)

**inf** Série de Seminários do INF  
227 subscribers

SUBSCRIBED [Unsubscribe](#)

# Podcasts



**Fronteiras da Engenharia de Software**

@FronteirasES



Episódio #8 no ar!

Arquitetura de Software com Ingrid Nunes (UFRGS)

[Translate Tweet](#)

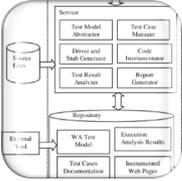
EP. 8  
Arquitetura de Software

#8. Arquitetura de Software com Ingrid Nunes (UFRGS) - Fronteiras da Engenharia de Software

[open.spotify.com](https://open.spotify.com)

com Ingrid Nunes (UFRGS)  
@ingridnunesIN  
<https://www.instagram.com/fronteirasES>

# Key Points



Have an architectural model



Do not neglect modularisation and separation of concerns



Adopt code review (and check the points above!)



Manage technical debt (and make payments!)



Use metrics and static analysis tools (automation!)



Software logging (with guidelines and standards!)

**Prof. Ingrid Nunes**  
UFRGS

Homepage  
[ingridnunes.github.io](https://ingridnunes.github.io)

Twitter  
[twitter.com/ingridnunesIN](https://twitter.com/ingridnunesIN)

Facebook  
[facebook.com/ingridnunesIN](https://facebook.com/ingridnunesIN)

LinkedIn  
[linkedin.com/in/ingrid-nunes](https://linkedin.com/in/ingrid-nunes)