APRESENTAÇÃO
DE APOIO

TÉCNICAS ÁGEIS DE PROGRAMAÇÃO

Guilherme Lacerda – Aula 01

PUCRS online uol edtech

Professores

DANIEL WILDT
Professor Convidado

Profissional de tecnologia preocupado com desenvolvimento de produtos e serviços com equipes focadas em aprendizado, melhoria contínua e autonomia. Mentora e produz conteúdo em vídeo, áudio e texto sobre: consciência de tempo, experiência de usuário, empreendedorismo e metodologias ágeis. Sócio e mentor na Wildtech, Blogger/Youtuber no danielwildt.com, sócio e diretor na uMov.me.

GUILHERME LACERDA
Professor Convidado

Graduado em Informática pela Universidade da Região da Campanha (2000). Mestre em Ciência da Computação pela Universidade Federal do Rio Grande do Sul (2005). Atualmente, cursa Doutorado em Ciência da Computação na UFRGS, na área de Engenharia de Software. Consultor/Instrutor associado da Wildtech, trabalhando com coaching e mentoring nas áreas de Engenharia de Software, Gerência de Projetos e Produtos e Metodologias Ágeis (eXtreme Programming, SCRUM, Lean). Possui mais de 20 anos de experiência em desenvolvimento de software. Atuou por vários anos como analista/projetista/desenvolvedor de software. Possui as certificações de SCRUM Master (SCM) e SCRUM Professional (CSP) pela SCRUM Alliance. Membro do IASA (International Association of Software Architects). Fundador do Grupo de Usuários de Métodos Ágeis (GUMA), vinculado a SUCESU-RS. É docente de graduação (Ciência da Computação, Análise e Desenvolvimento de Sistemas e Sistemas de Informação, Gestão de TI - Unisinos) e pós-graduação (Engenharia de Software, Desenvolvimento de Aplicações Móveis - Unisinos e Desenvolvimento Full Stack - PUCRS).

Professores

MICHAEL DA COSTA MÓRA
Professor PUCRS

Graduado em Ciência da Computação pela Universidade Federal do Rio Grande do Sul (UFRGS), mestre em Computação e doutor em Ciência da Computação pela mesma universidade. Professor-adjunto do Instituto de Informática. Tem experiência na área de ciência da computação com ênfase em inteligência artificial, atuando principalmente nos seguintes temas: inteligência artificial, aprendizagem de máquina, agentes inteligentes e sistemas multiagentes, engenharia de software e desenvolvimento de sistemas, ensino de programação e de ciência da computação.

Ementa da disciplina

Fundamentos da agilidade: primórdios, manifesto ágil, princípios da agilidade. Panorama das metodologias ágeis. Extreme programming: características, valores, práticas, as práticas na prática. Test driven development (TDD): origens, codificar – testar – projetar, benefícios e armadilhas, variações, TDD na prática. Behaviour driven design (BDD): origens e princípios, BDD x TDD, benefícios e armadilhas, BDD na prática.

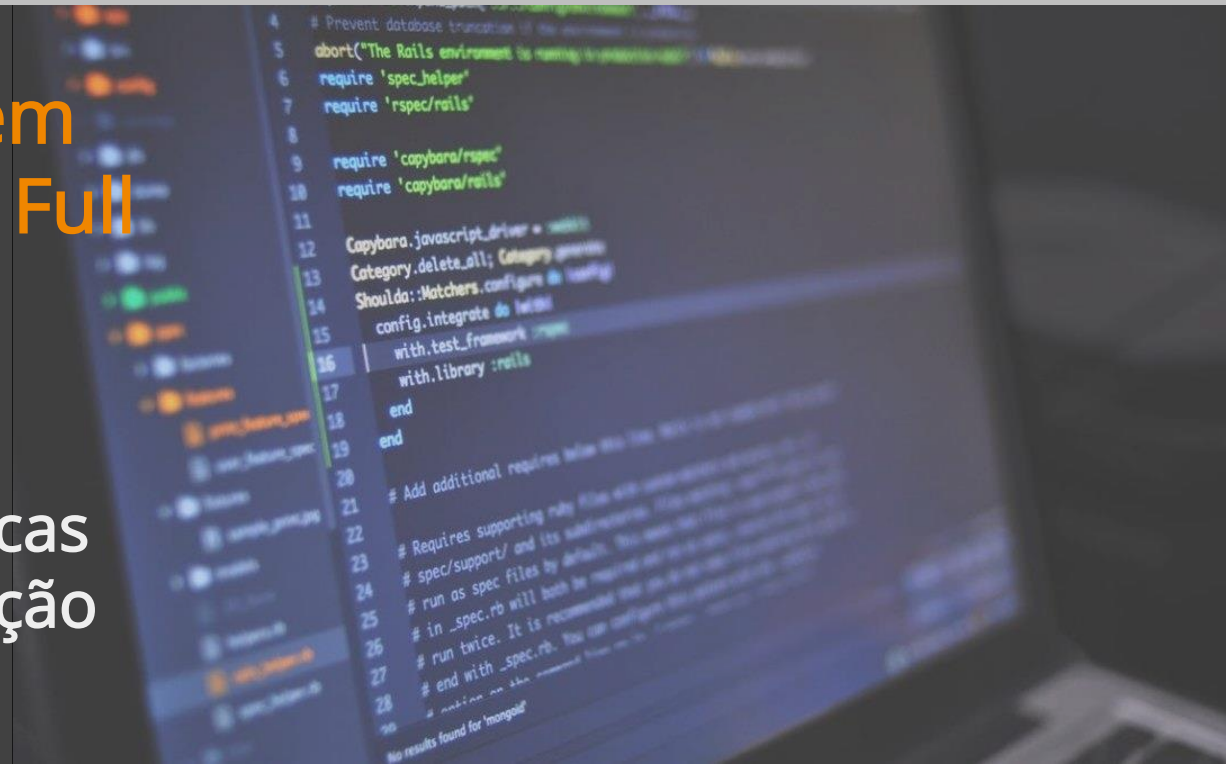
PUCRS

uol edtech

WILDTECH
start wild, keep wild

Especialização em Desenvolvimento Full Stack

Disciplina de Técnicas Ágeis de Programação



Parte 3

Guilherme Lacerda

- SW Engineering Lead (umbler)
- Mentor (wildtech)
- Professor Universitário (unisinos)
- site | twitter

Agenda - Introdução
Parte 3

1. Metáfora
2. Trabalho Colaborativo

“Todas as religiões, artes e ciências são ramos da mesma árvore.”

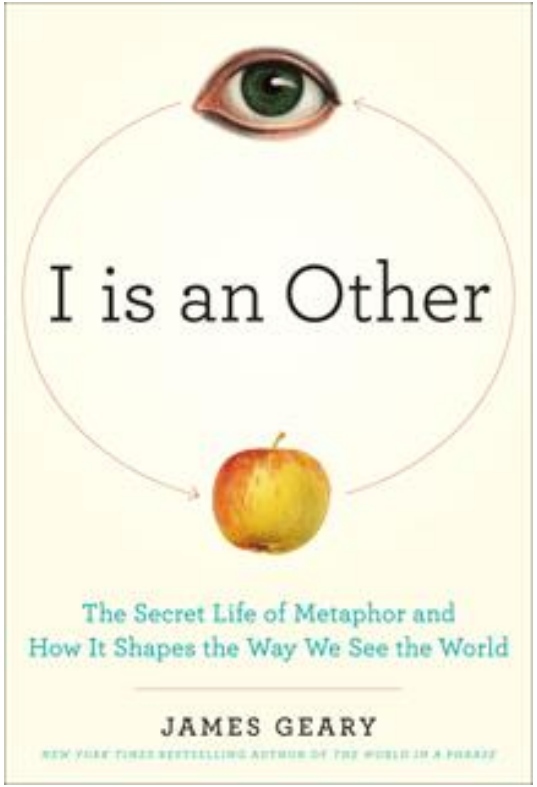
Albert Einstein

O que é Metáfora?

O Uso

- A metáfora não é só uma figura de linguagem

- Ela está presente de forma intensa porém imperceptível em tudo que os humanos fazem



O Uso

- Nós falamos uma metáfora a cada 10 a 25 palavras
- Ou seja, cerca de seis metáforas por minuto
- A metáfora é uma maneira de pensar muito antes de ser um estilo com palavras

Metáforas no desenvolvimento de software

THE EVOLUTION OF

SOFTWARE ARCHITECTURE

1990's

SPAGHETTI-ORIENTED
ARCHITECTURE
(aka Copy & Paste)

2000's

LASAGNA-ORIENTED
ARCHITECTURE
(aka Layered Monolith)

2010's

RAVIOLI-ORIENTED
ARCHITECTURE
(aka Microservices)

WHAT'S NEXT?

PROBABLY PIZZA-ORIENTED ARCHITECTURE

By @benorama

Por que elas são importantes?

- As metáforas têm um **grande valor**
- Ampliam o **poder de comunicação**
- Compreensão **compartilhada e colaborativa**
- Uso **de alto e baixo nível, diferentes níveis de abstração**

“O talento vence jogos, mas só o trabalho em equipe vence campeonatos.”

Michael Jordan

O trabalho da pessoa desenvolvedora

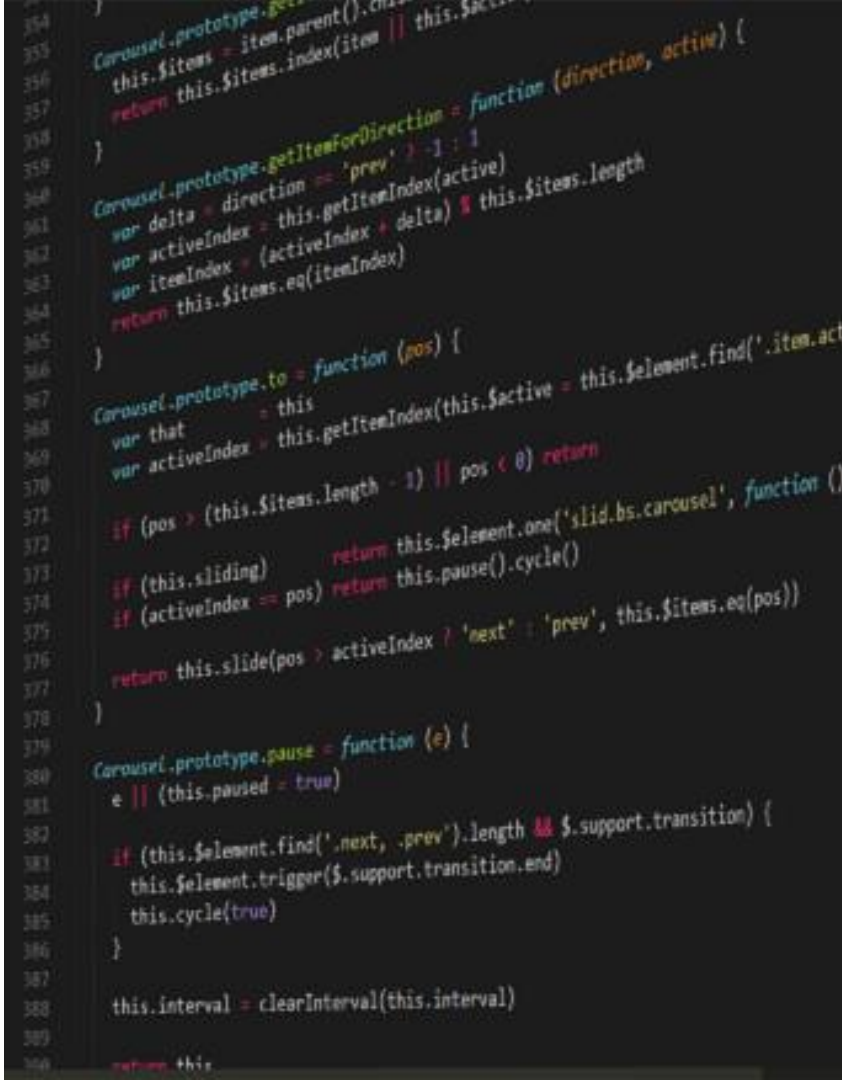
- Principais Atividades
  - Design/Programação/Testes/Manutenção/Evolução
  - Gestão de configuração e do trabalho
  - Colaborar com outros profissionais

- Desafios
  - Dominar tecnologia(s)
  - Manter-se atualizada(o)
  - Ter a visão do todo
  - Ser especialista/generalista
  - Interagir com clientes/usuários
  - Preocupar-se constantemente com a qualidade
  - Se adaptar!

Padrões/Convenções

- Toda a linguagem tem
  - E por que não usamos?
  - Quais são os benefícios?
  - Prática do XP

- Componentes
  - Nomenclatura
  - Estrutura do código
  - Terminologia
  - Formatação
  - Boas Práticas
  - Exemplos



Pair Programming e Mob Programming

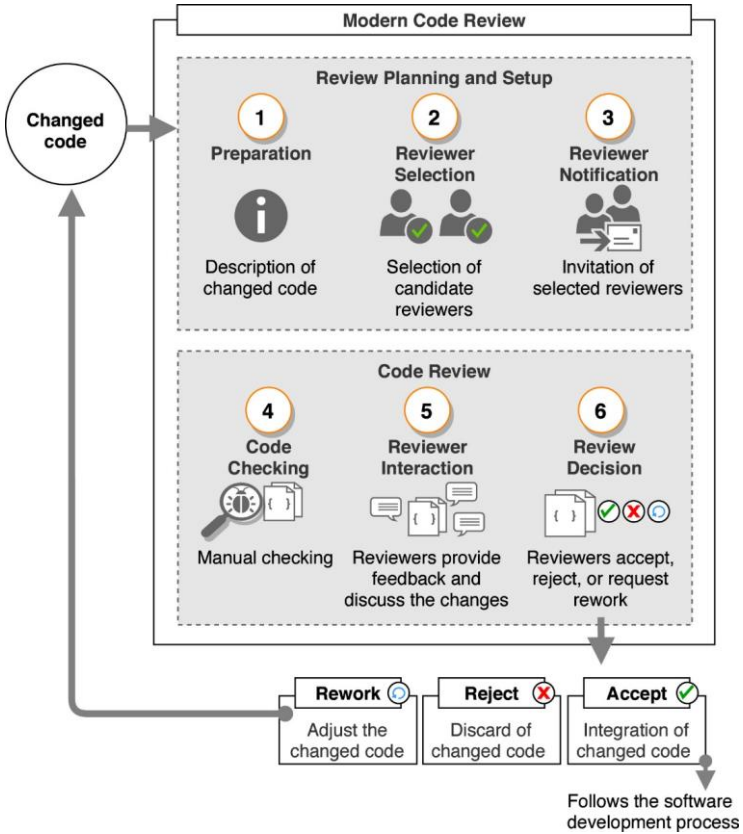
- **Pair Programming**
  - Mecânica
  - Ferramentas
- **Mob Programming**
  - Mecânica

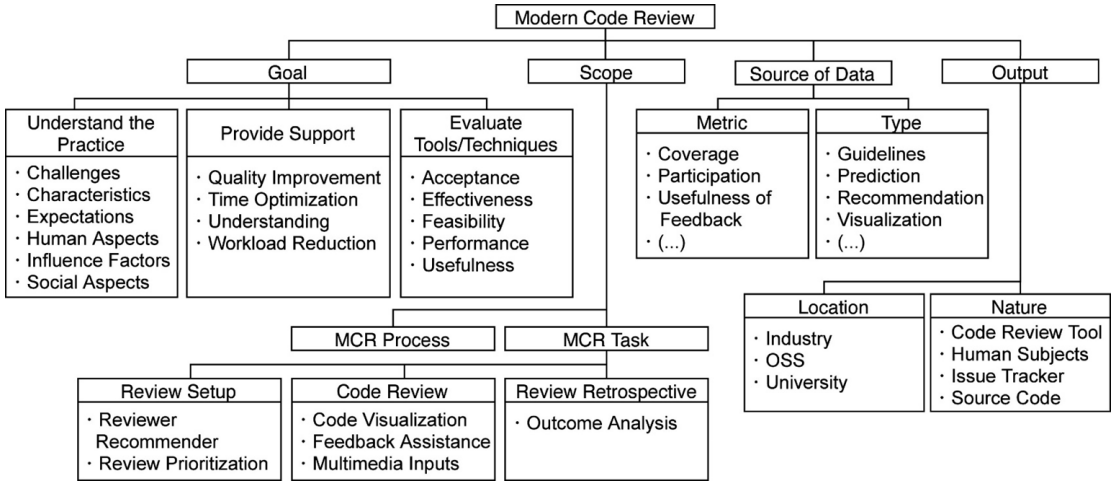
e

Patterns

github.com/michaelkeeling/mob-programming-patterns

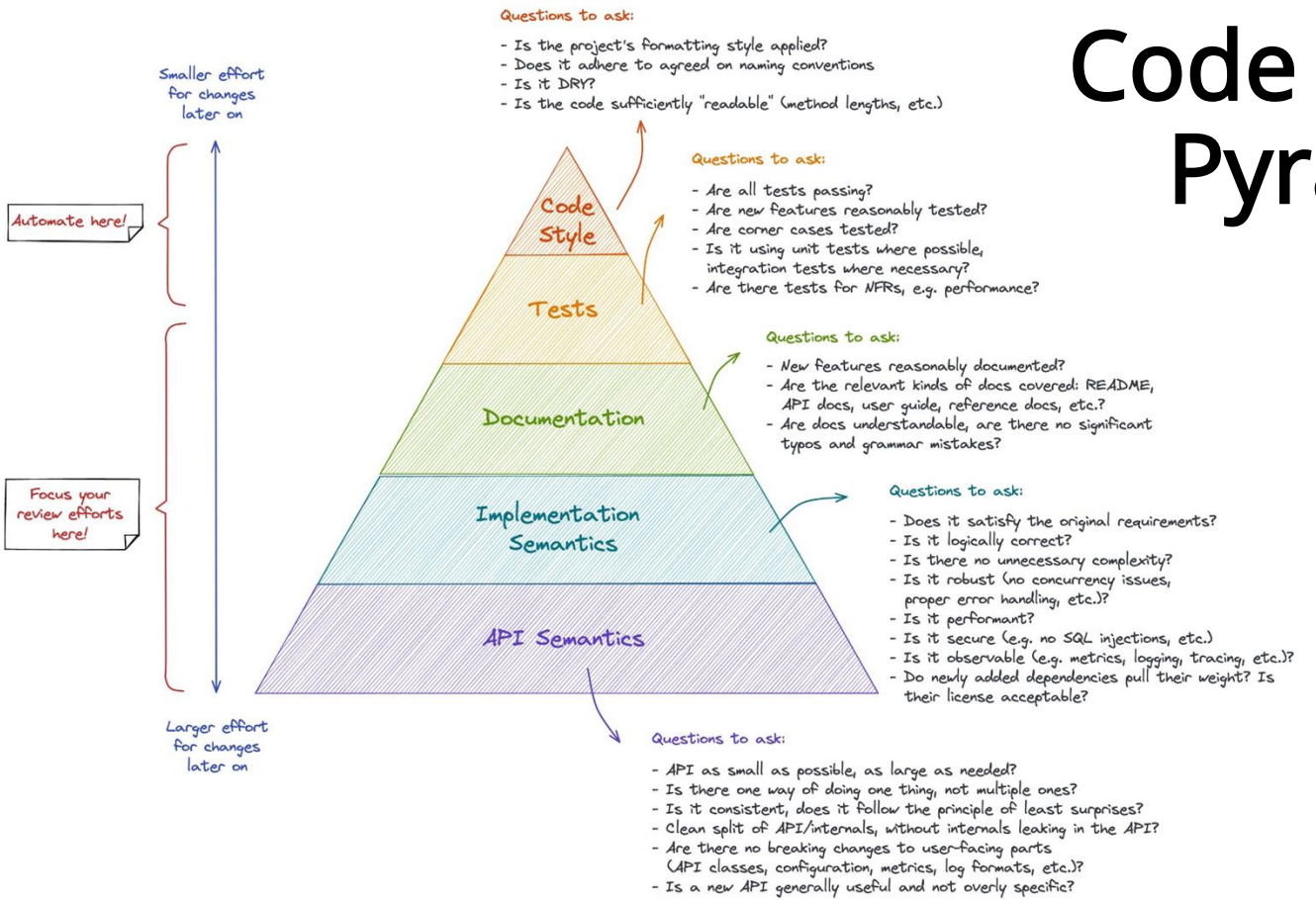
Modern Code Review





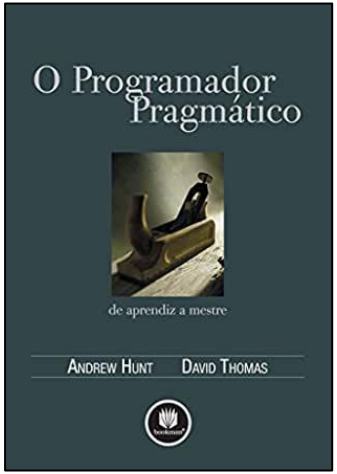
Davila N. & Nunes I. (2021) A systematic literature review and taxonomy of modern code review. Journal of Systems and Software (177).

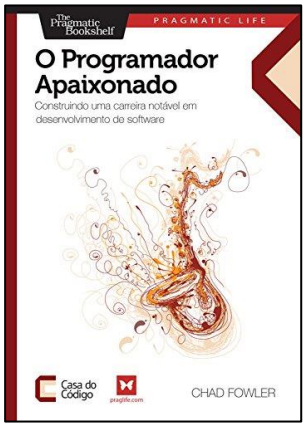
Code Review Pyramid

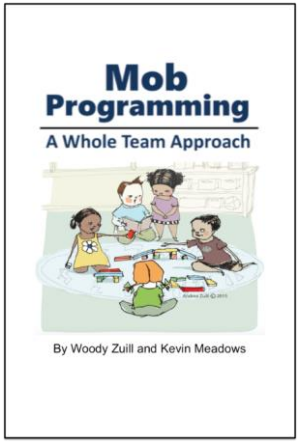


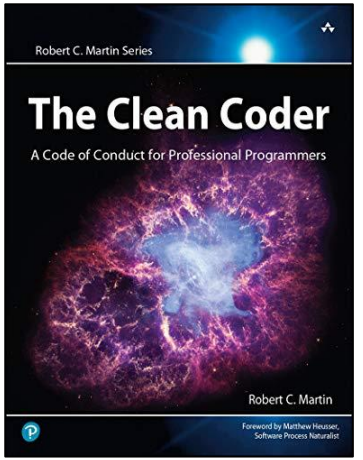
Gunnar Morling - twitter.com/gunnarmorling/status/1501645187407388679

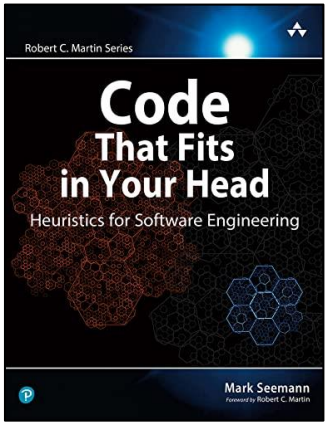
Para aprofundar os estudos...

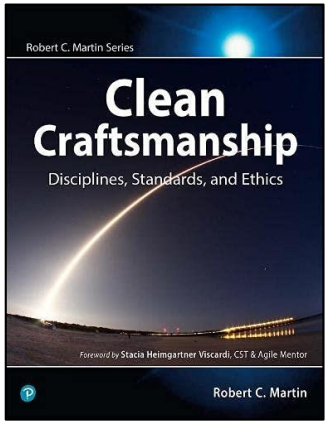












Fim Parte 3

Parte 4

Agenda - Introdução
Parte 4

1. Smells
2. Manutenção e Evolução de Código

O que é um código ruim?

Bad Smells

Mas quando começou a se falar sobre smells?

Contents lists available at ScienceDirect

The Journal of Systems and Software

journal homepage: www.elsevier.com/locate/jss

Code smells and refactoring: A tertiary systematic review of challenges and observations

Guilherme Lacerda[a,b,*], Fabio Petrillo[c], Marcelo Pimenta[b], Yann Gaël Guéhéneuc[d]

[a] University of Vale do Rio dos Sinos, Polytechnic School São Leopoldo, RS, Brazil
[b] Federal University of Rio Grande do Sul, Institute of Informatics Porto Alegre, RS, Brazil
[c] University of Quebec at Chicoutimi, Department of Computer Science & Mathematics Chicoutimi, Quebec, Canada
[d] Concordia University, Departement of Computer Science and Software Engineering Montreal, Quebec, Canada

ARTICLE INFO

Article history:
Received 11 September 2019
Revised 25 February 2020
Accepted 18 April 2020
Available online 30 April 2020

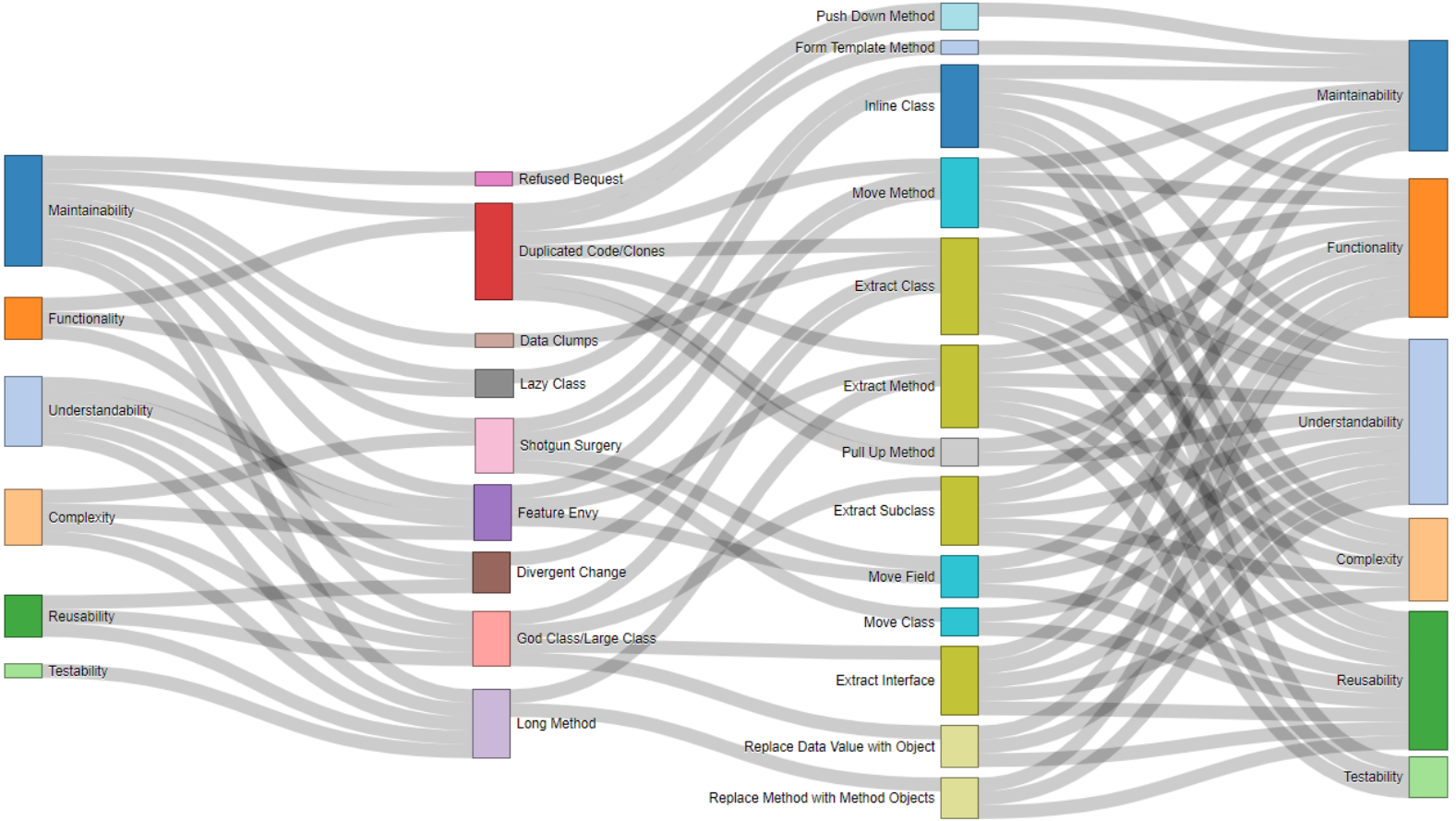
Keywords:
Code smells
Refactoring
Tertiary systematic review

ABSTRACT

Refactoring and smells have been well researched by the software-engineering research community these past decades. Several secondary studies have been published on code smells, discussing their implications on software quality, their impact on maintenance and evolution, and existing tools for their detection. Other secondary studies addressed refactoring, discussing refactoring techniques, opportunities for refactoring, impact on quality, and tools support.

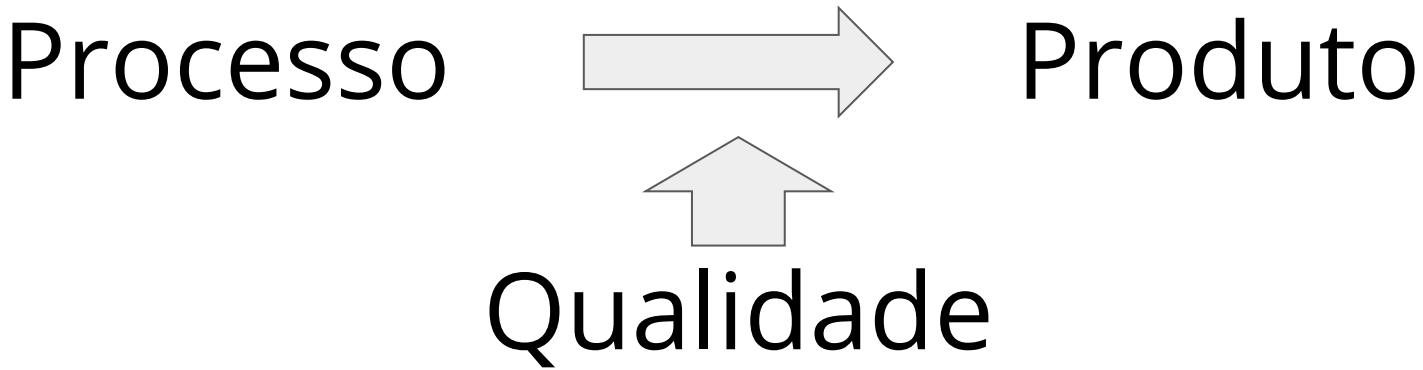
In this paper, we present a tertiary systematic literature review of previous surveys, secondary systematic literature reviews, and systematic mappings. We identify the main observations (*what we know*) and challenges (*what we do not know*) on code smells and refactoring. We perform this tertiary review using eight scientific databases, based on a set of five research questions, identifying 40 secondary studies between 1992 and 2018.

Lacerda G., Petrillo F., Pimenta M. & Guéhéneuc Y. (2020). **Code smells and refactoring: A tertiary systematic review of challenges and observations**, Journal of Systems and Software, Volume 167.



Lacerda G., Petrillo F., Pimenta M. & Guéhéneuc Y. (2020). **Code smells and refactoring: A tertiary systematic review of challenges and observations**, Journal of Systems and Software, Volume 167.

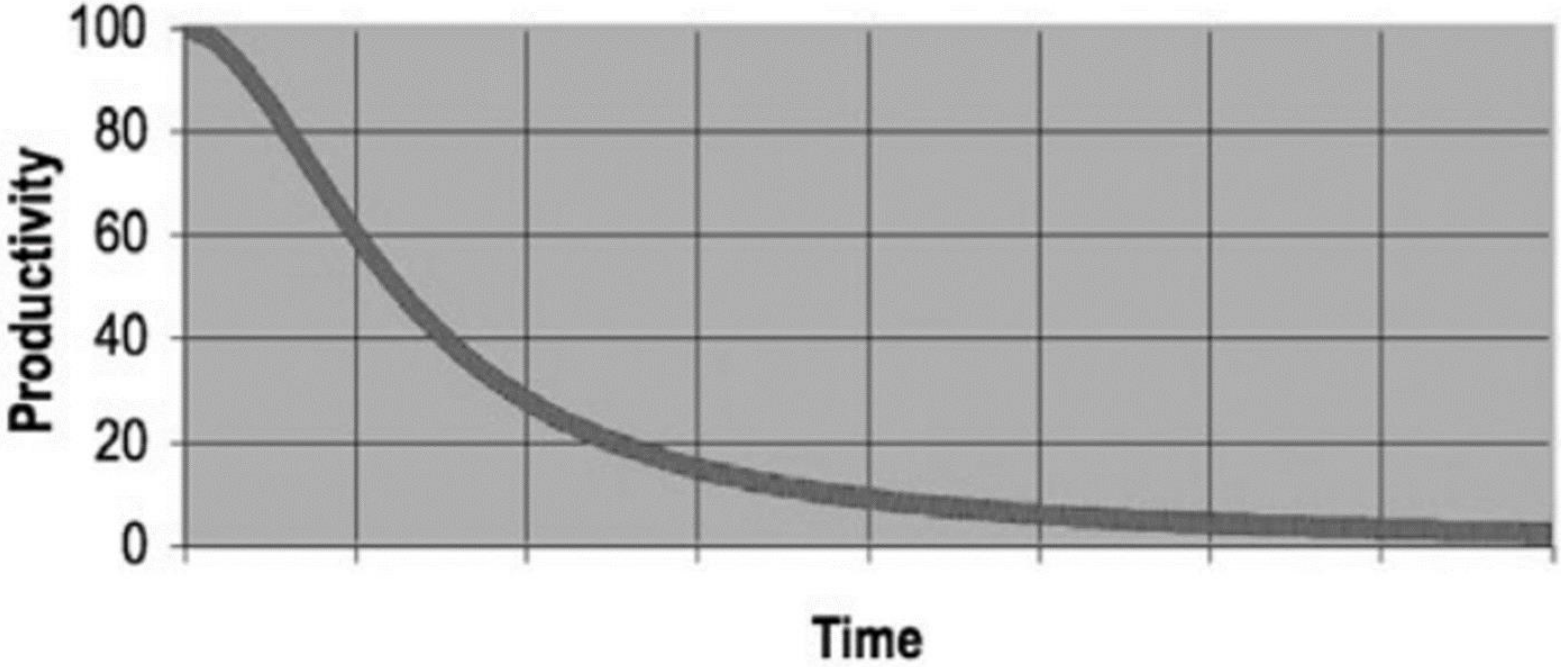
Engenharia de Software



…e quando o software ficar pronto…

É fundamental entender a natureza do software
(produto e processo)

Evolução



Robert Martin, *Clean Code: A handbook of agile software craftsmanship, Prentice Hall PTR (2008)* (extra *artigo infoq*)

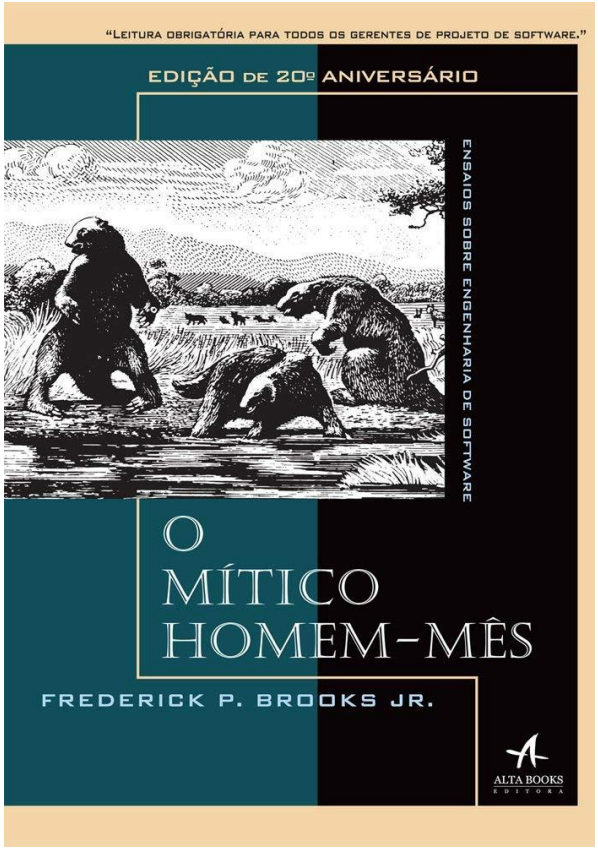
Negócio e TI

Complexidade

Essencial

X

Acidental



Leis de Lehman

Lei	Descrição
Mudança contínua	Um programa usado em um ambiente do mundo real deve necessariamente mudar, ou se torna progressivamente menos útil nesse ambiente.
Aumento da complexidade	Como um programa em evolução muda, sua estrutura tende a tornar-se mais complexa. Recursos extras devem ser dedicados a preservar e simplicar a estrutura.
Evolução de programa de grande porte	A evolução de programa é um processo de autorregulação. Atributos de sistema como tamanho, tempo entre *releases* e número de erros relatados são aproximadamente invariáveis para cada *release* do sistema.
Estabilidade organizacional	Ao longo da vida de um programa, sua taxa de desenvolvimento é aproximadamente constante e independente dos recursos destinados ao desenvolvimento do sistema.
Conservação da familiaridade	Durante a vigência de um sistema, a mudança incremental em cada *release* é aproximadamente constante.
Crescimento contínuo	A funcionalidade oferecida pelos sistemas tem de aumentar continuamente para manter a satisfação do usuário.
Declínio de qualidade	A qualidade dos sistemas cairá, a menos que eles sejam modificados para refletir mudanças em seu ambiente operacional.
Sistema de *feedback*	Os processos de evolução incorporam sistemas de *feedback* multiagentes, *multiloop*, e você deve tratá-los como sistemas de *feedback* para alcançar significativa melhoria do produto.

leis de Lehman (wikipedia)

Manutenção

Table 5.1. Software Maintenance Categories		
	Correction	**Enhancement**
Proactive	Preventive	Perfective
Reactive	Corrective	Adaptive

SWEBoK v3.0, *IEEE Computer Society (2014)*

Visões

Desenvolvimento + Manutenção

=

Evolução!

Problemas na Documentação

- Taxonomia
  - Construída a partir de mais de 800 documentos, incluindo PRs do Github e threads do StackOverflow
  - Problemas: Incompletude, falta de atualização, pouca usabilidade, pouca legibilidade

- Principais dores
  - Documentações que ferem os critérios de completude, atualidade e legibilidade
  - Falta de Corretude: documentações com informações erradas

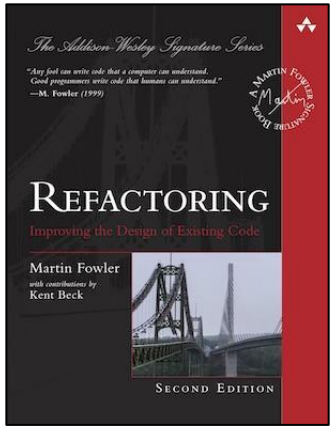
Aghajani E. et al. (2019) **Software Documentation Issues Unveiled**. In: Proceedings of 41st ACM/IEEE International Conference on Software Engineering (ICSE 2019)
Aghajani E. et al. (2020) **Software Documentation: The Practitioner´s Perspective**. In: Proceedings of 42nd ACM/IEEE International Conference on Software Engineering (ICSE 2020)

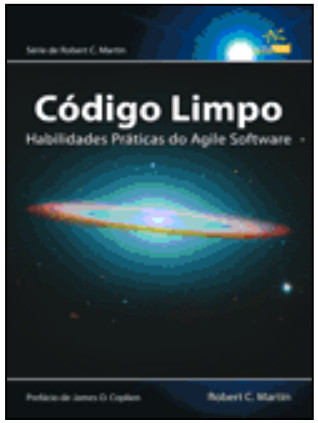
Documentação Técnica

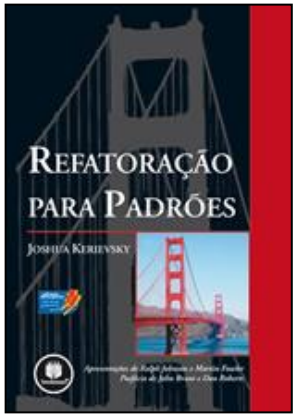
- Estrutura
  - Conteúdo, tamanho, audiência, formato (tutorial, vídeos)
  - Contar uma história
  - Use e abuse de exemplos de código

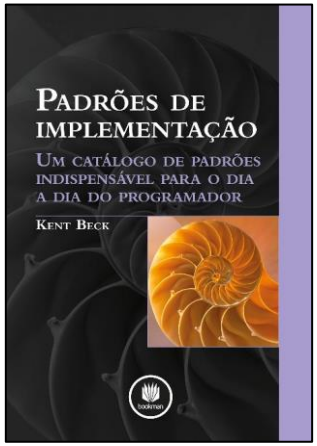
- Elementos e exemplos
  - Diagramas (Maps, UML, entre outros)
  - Arquitetura: C4 Model, Architecture Haiku, Architecture Decision Records (ADR)
  - Mockups, protótipos navegáveis
  - Engenharia Reversa: Artefatos, Visualização de Software
  - Código: Programação Literata, comentários e documentações no código
  - Testes: documentação automatizada
  - Guias: Deployment, contribuição, instalação/uso, primeiros passos

Para aprofundar os estudos...









Fim Parte 4


PUCRS online
uol edtech