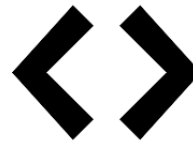
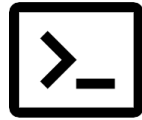
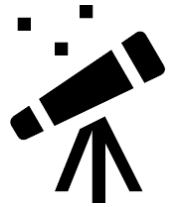


decodificando o

# CODE REVIEW





# Elaine Naomi Watanabe

Desenvolvedora de Software (Plataformatec)

Mestrado em Ciência da Computação (USP)



*[twitter.com/elaine\\_nw](https://twitter.com/elaine_nw)*



*[speakerdeck.com/elainenaomi](https://speakerdeck.com/elainenaomi)*

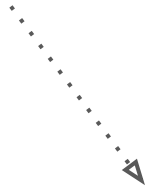


[careers.plataformatec.com.br](https://careers.plataformatec.com.br)

## expectativas:

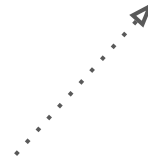
discutir os *desafios* e *práticas* da  
revisão de código

definição



práticas do  
dia a dia

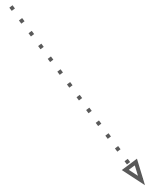
desafios



aprendizados

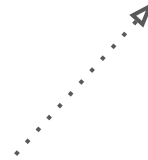


definição



práticas do  
dia a dia

desafios



aprendizados



# CODE REVIEW

processo de verificação de um sistema por meio da **análise do código fonte**, realizada por humanos

[https://en.wikipedia.org/wiki/Code\\_review](https://en.wikipedia.org/wiki/Code_review)

# CODE REVIEW

processo de verificação  
análise do código fonte

[https://en.wikipedia.org/wiki/Code\\_r](https://en.wikipedia.org/wiki/Code_review)





qual é o objetivo?



# Software Defect Reduction Top 10 List

Barry Boehm, University of Southern California  
Victor R. Basili, University of Maryland

**R**ecently, a National Science Foundation grant enabled us to insight has been a major driver in focusing industrial software practice on thor-

## TWO

*Current software projects spend about 40 to 50 percent of their effort on avoidable rework.*

Such rework consists of effort spent fixing software difficulties that could have been discovered earlier and fixed less expensively or avoided altogether. By implication, then, some effort must consist of “unavoidable rework,” an observation that has gained increasing credibility with the growing realization that better user-interactive systems result from *emergent* processes. In such processes, the requirements emerge from prototyping and other multistakeholder-shared learning activities, a departure from traditional *reductionist* processes that stipulate requirements in advance, then reduce them to practice via design and coding. Emergent processes indicate

60% dos defeitos podem ser  
identificados na revisão do código

Boehm, Barry, and Victor R. Basili. "Top 10 list [software development]."

*Computer* 34.1 (2001): 135-137

# What Types of Defects Are Really Discovered in Code Reviews?

Mika V. Mäntylä and Casper Lassenius, *Member, IEEE*

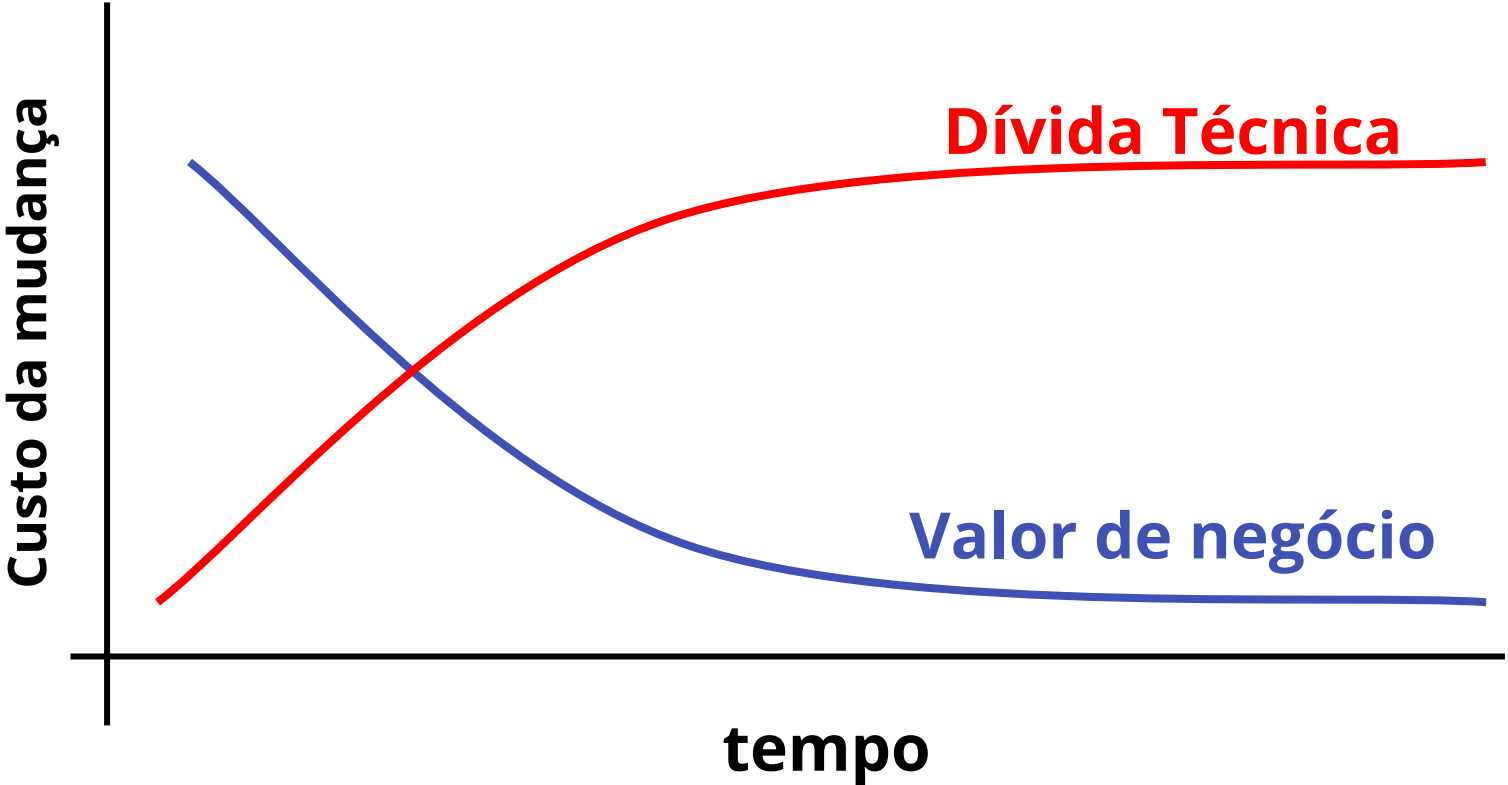
**Abstract**—Research on code reviews has often focused on defect counts instead of defect types, which offers an imperfect view of code review benefits. In this paper, we classified the defects of nine industrial (C/C++) and 23 student (Java) code reviews, detecting 388 and 371 defects, respectively. First, we discovered that 75 percent of defects found during the review do not affect the visible functionality of the software. Instead, these defects improved software evolvability by making it easier to understand and modify. Second, we created a defect classification consisting of functional and evolvability defects. The evolvability defect classification is based on the defect types found in this study, but, for the functional defects, we studied and compared existing functional defect classifications. The classification can be useful for assigning code review roles, creating checklists, assessing software evolvability, and building software engineering tools. We conclude that, in addition to functional defects, code reviews find many evolvability defects and, thus, offer additional benefits over execution-based quality assurance methods that cannot detect evolvability defects. We suggest that code reviews may be most valuable for software products with long life cycles as the value of discovering evolvability defects in them is greater than for short life cycle systems.

**Index Terms**—Code inspections and walkthroughs, enhancement, extensibility, maintainability, restructuring.



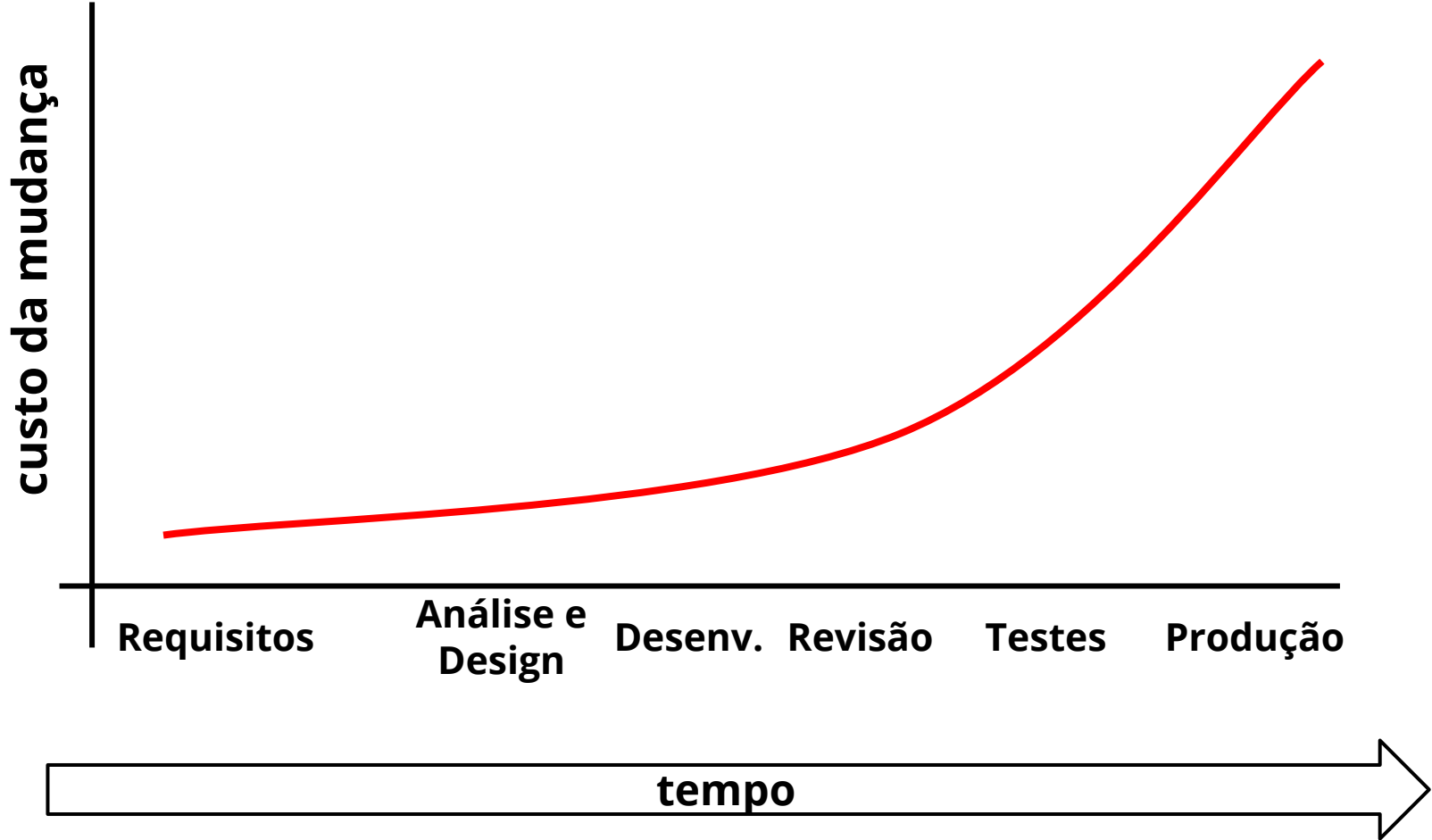
Revisão de código é uma boa ferramenta para identificar defeitos relacionados à **evolutibilidade** do código que não são identificáveis na fase de testes

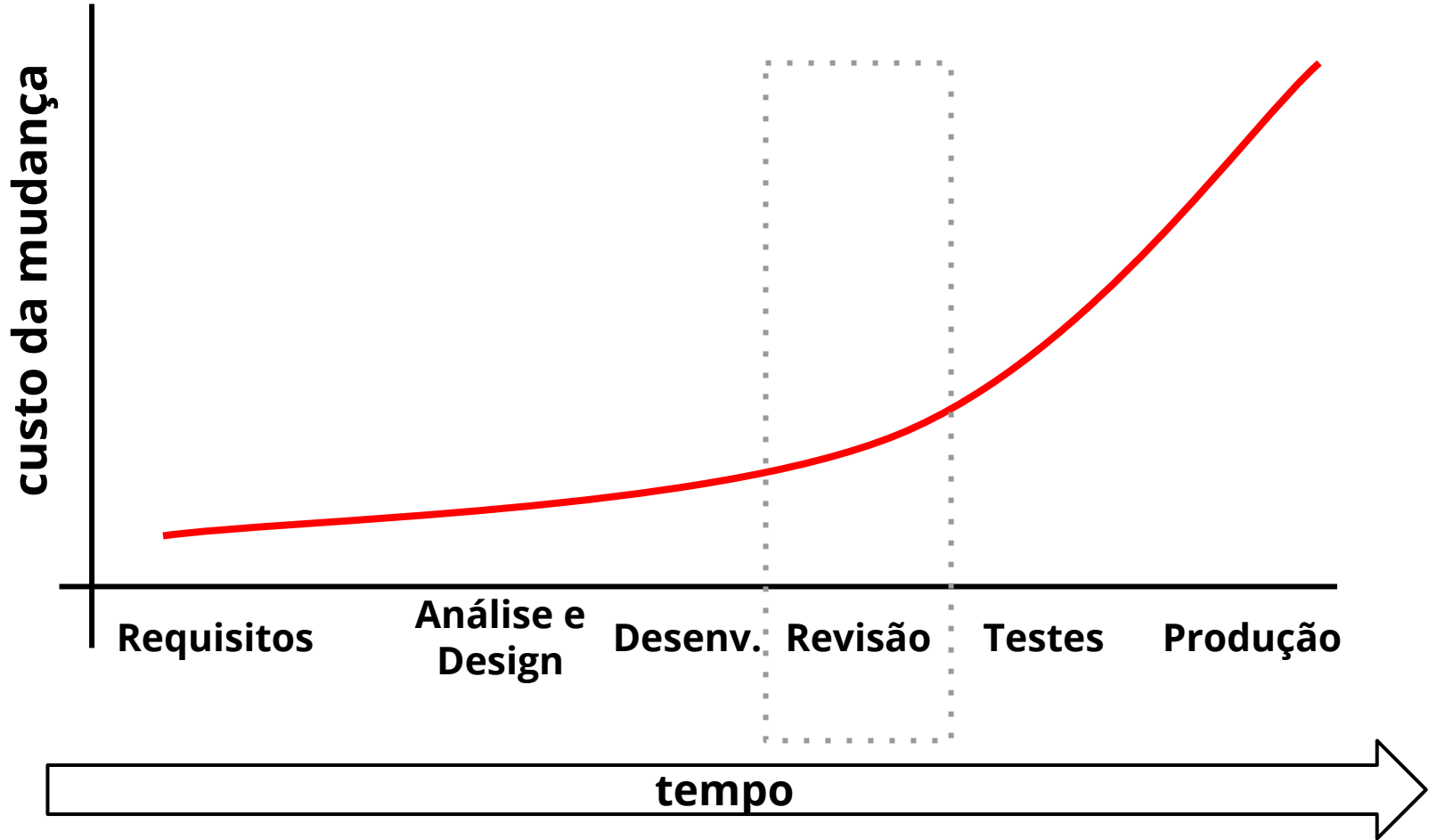
Mäntylä, Mika V., and Casper Lassenius. "What types of defects are really discovered in code reviews?." *IEEE Transactions on Software Engineering* 35.3 (2009): 430-448











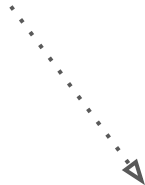
# QUALIDADE DE SOFTWARE

- ✓ Confiabilidade
- ✓ Corretude
- ✓ Eficiência
- ✓ Manutenibilidade
- ✓ Valor de negócio

[https://en.wikipedia.org/wiki/Software\\_quality](https://en.wikipedia.org/wiki/Software_quality)



definição



desafios



práticas do  
dia a dia



aprendizados

como fazer um code review?



File View Bookmarks Tools

TC <http://www.techcrunch.com/2005/11/22/wordpress-20-the-great>

TC Techcrunch » Archives

Weblog Tools Collection

TC Techcrunch » Blog ...

*This is a welcome feature.*

*Also added a wysiwyg rich text editor. I disabled  
ately, although many users will like it.*

✓ Pair Programming

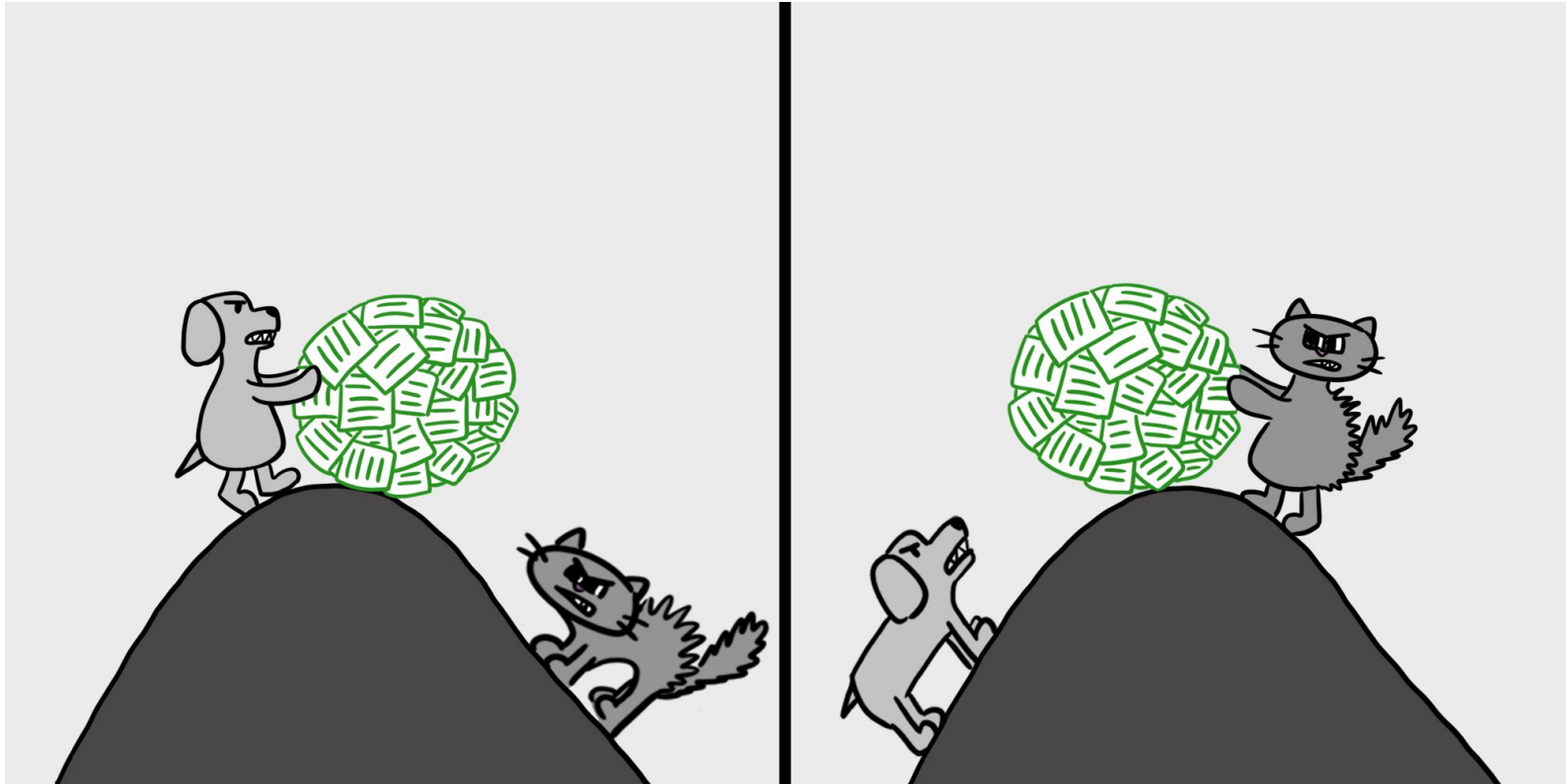
✓ Pull Request



✓ Pair Programming

✓ Pull Request

Fonte: <https://mtlynch.io/human-code-reviews-2/>





✓ Pair Programming

✓ Pull Request

**código + contexto de negócio**

✓ Pair Programming

✓ Pull Request

**histórico acessível das discussões**

	Mesmo tempo	Tempo diferente
Locais diferentes	interação síncrona distribuída	interação assíncrona distribuída
Mesmo local	interação face-a-face	interação assíncrona

Johansen, Robert. "Groupware: Future directions and wild cards."  
*Journal of Organizational Computing and Electronic Commerce* 1.2 (1991): 219-227.

# [pt] Fix a Portuguese typo in the `basic/control-structures.md` #1869

Edit

**Merged** anderkonzen merged 2 commits into `elixirschool:master` from `elainenaomi:ew-pt-fix-typo` on May 24

Conversation 1

Commits 2

Checks 0

Files changed 1

+2 -2



elainenaomi commented on May 24

Contributor



Hi, everyone! It's my first contribution.

It's just a minor typo fixing.

I'm considering the spelling in pt-BR, in which, the word `encadiar` is a typo (the correct spelling is `encadear`). I'm not sure if this word is correct in other variants from Portuguese.

Thanks for your attention.

Reviewers



anderkonzen



thiamsantos



Assignees

No one assigned

Labels

translation



elainenaomi added some commits on May 24

revisão por meio de comentários





thiamsantos approved these changes on May 24

[View changes](#)



anderkonzen added the **translation** label on May 24




anderkonzen approved these changes on May 24

[View changes](#)

anderkonzen left a comment

Member



Thank you very much @elainenaomi! 🙌  TOP  
Any improvements, small or big, make the difference!



anderkonzen merged commit **01e84a2** into `elixirschool:master`  
on May 24  
1 check passed

[View details](#)

[Revert](#)

No milestone

Notifications

[Customize](#)

 [Unsubscribe](#)

You're receiving notifications because you were mentioned.

3 participants



[Allow edits from maintainers.](#)

[Learn more](#)

transferência de conhecimento

mentoria

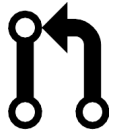
visibilidade das alterações  
para outros times

team awareness

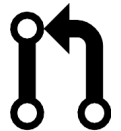
boas práticas

como pessoa autora

- ✓ Título explicativo
- ✓ **Motivação** (contexto de negócio)
- ✓ Lista de dúvidas e discussões prévias
- ✓ Gifs, screenshots das alterações



- ✓ Mensagens de commits coerentes
- ✓ Código completo, testado
- ✓ Alterações pequenas
- ✓ **Single responsibility principle**



- ✓ Marcar pessoas como revisoras
- ✓ Aplicar as alterações necessárias
- ✓ **Responder** a todos os comentários







como pessoa revisora

- ✓ Identificar defeitos (bugs)
- ✓ Sugerir soluções alternativas, refatorações
- ✓ Reforçar padrões de código e design
- ✓ Validar funcionalidade (**código + negócio**)



- ✓ Identificar problemas de segurança
- ✓ Analisar impactos na performance
- ✓ Sugerir documentações
- ✓ Validar a **qualidade do código-fonte**



- ✓ Conhecer novas funcionalidades
- ✓ Aprender novas tecnologias
- ✓ Compartilhar conhecimento e dúvidas





definição



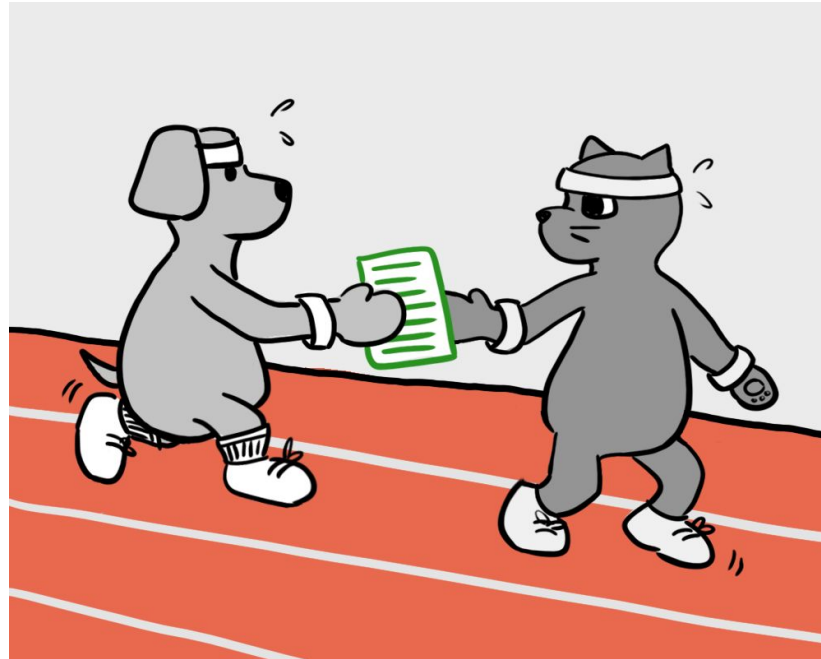
práticas do  
dia a dia

desafios



aprendizados

Fonte: <https://mtlynch.io/human-code-reviews-1/>





# Modern Code Review: A Case Study at Google

Caitlin Sadowski, Emma Söderberg,  
Luke Church, Michal Sipko  
Google, Inc.

{supertri,emso,lukechurch,sipkom}@google.com

Alberto Bacchelli  
University of Zurich  
bacchelli@ifi.uzh.ch

## ABSTRACT

Employing lightweight, tool-based code review of code changes (aka *modern code review*) has become the norm for a wide variety of open-source and industrial systems. In this paper, we make an exploratory investigation of modern code review at Google. Google introduced code review early on and evolved it over the years; our study sheds light on why Google introduced this practice and analyzes its current status, after the process has been refined through decades of code changes and millions of code reviews. By means of 12 interviews, a survey with 44 respondents, and the analysis of review logs for 9 million reviewed changes, we investigate motivations behind code review at Google, current practices, and developers' satisfaction and challenges.

An open research challenge is understanding which practices represent valuable and effective methods of review in this novel context. Rigby and Bird quantitatively analyzed code review data from software projects spanning varying domains as well as organizations and found five strongly convergent aspects [33], which they conjectured can be prescriptive to other projects. The analysis of Rigby and Bird is based on the value of a *broad* perspective (that analyzes multiple projects from different contexts). For the development of an empirical body of knowledge, championed by Basili [7], it is essential to also consider a *focused and longitudinal* perspective that analyzes a single case. This paper expands on work by Rigby and Bird to focus on the review practices and characteristics established at Google, *i.e.*, a company with a multi-decade history of code review and a high-volume of daily reviews.

70% das alterações do Google  
são integradas em menos de 24h  
após o pedido de review

Sadowski, Caitlin, et al. "Modern code review: a case study at Google." *Proceedings of the 40th International Conference on Software Engineering: Software Engineering in Practice*. ACM, 2018

# Alterações pequenas, uma pessoa revisora e sem comentários além de autorização para integração

Sadowski, Caitlin, et al. "Modern code review: a case study at Google." *Proceedings of the 40th International Conference on Software Engineering: Software Engineering in Practice*. ACM, 2018

mágica?



alinhamento

Sua base de código parece ter sido  
escrita por **ÚNICA PESSOA?**

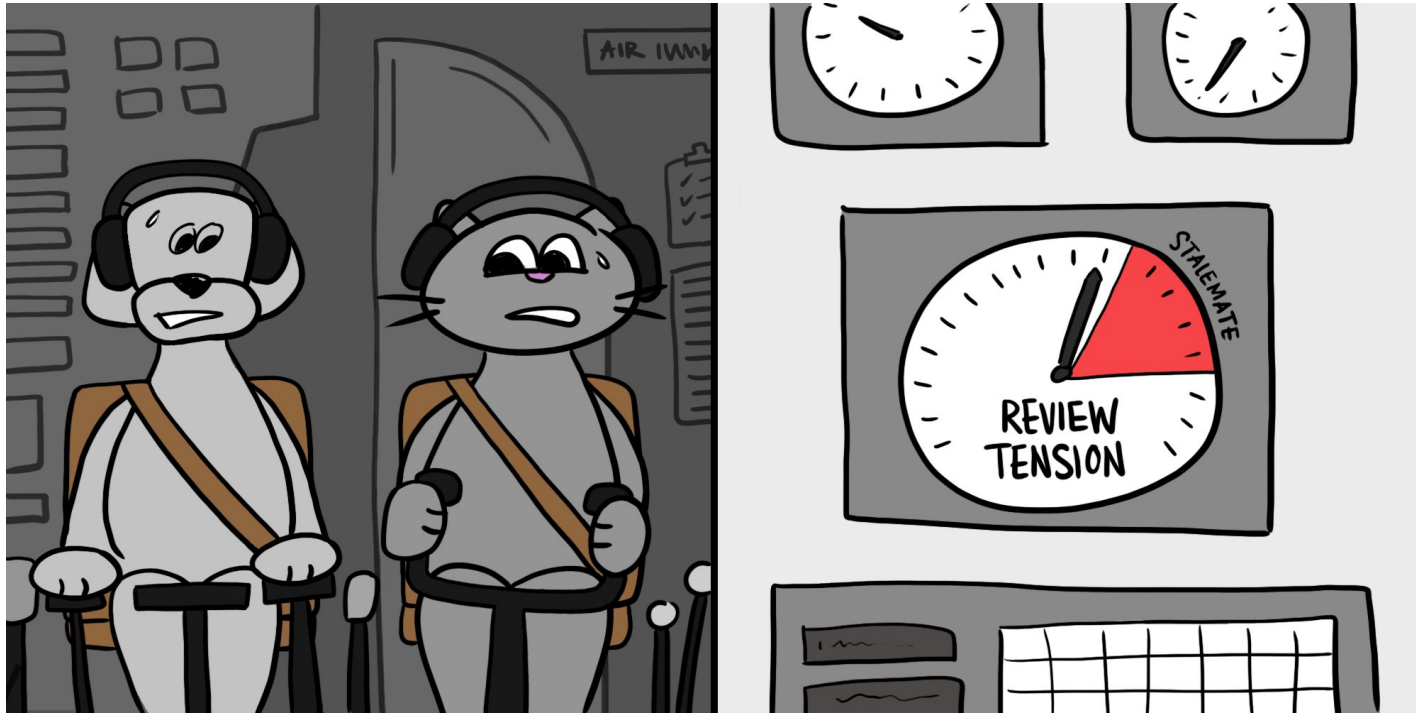
SIM

NÃO

# collective code ownership

<https://martinfowler.com/bliki/CodeOwnership.html>

Fonte: <https://mtlynch.io/human-code-reviews-2/>





quem faz review, faz parte da  
construção da solução também

como ir nessa direção?



como pessoa



lembre-se que o feedback  
deve ser **sobre o código,**  
e não sobre as pessoas

ninguém acorda e pensa:  
vou lá adicionar um bug e já volto





apoie a participação de

**TODAS AS PESSOAS** do seu time

não é porque alguém é experiente,  
que não vai errar

não é porque alguém é iniciante,  
que não vai ter contribuição



use comentários  
explícitos e descritivos

8

9 # Bundle edge Rails instead: gem 'rails', github: 'rails/rails'

10 +gem 'rails', '~> 5.1.6'



**random-person-123** just now



Reply...

```
8
9 # Bundle edge Rails instead: gem 'rails', github: 'rails/rails'
10 +gem 'rails', '~> 5.1.6'
```

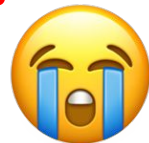


random-person-123 just now



Reply...

**é para eu jogar fora a minha alteração?**



```
8
9 # Bundle edge Rails instead: gem 'rails', github: 'rails/rails'
10 +gem 'rails', '~> 5.1.6'
```



random-person-123 just now



Reply...

**ah, era só para apagar o espaço extra**

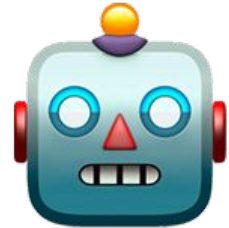


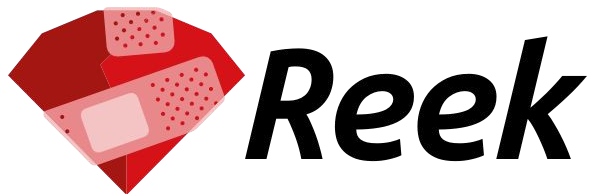




comentários repetitivos  
sobre estilo de código

podem ser substituídos por uma  
ferramenta de análise de código









**CÓDIGO: ELAINE**  
50% por 6 meses



melhorias de design podem ser entregues em outro pull request





se chegar a uma conclusão  
estiver difícil

não se limite à ferramenta de review

- ✓ videoconferência
- ✓ presencialmente

<http://blog.plataformatec.com.br/2018/11/trabalhando-com-times-distribuidos/>

**documente** as decisões e  
discussões offline





preste atenção na sua  
forma de se **comunicar**

muitas vezes não é óbvio que um comentário ou comportamento é prejudicial







como organização



tenha critérios bem definidos  
ex.: o número mínimo de aprovações



fator social

# Influence of Social and Technical Factors for Evaluating Contribution in GitHub

Jason Tsay, Laura Dabbish, James Herbsleb

School of Computer Science and Center for the Future of Work, Heinz College

Carnegie Mellon University

5000 Forbes Ave., Pittsburgh, PA 15213 USA

{jtsay, dabbish, jdh}@cs.cmu.edu

## ABSTRACT

Open source software is commonly portrayed as a meritocracy, where decisions are based solely on their technical merit. However, literature on open source suggests a complex social structure underlying the meritocracy. Social work environments such as GitHub make the relationships between users and between users and work artifacts transparent. This transparency enables developers to better use information such as technical value and social connections when making work decisions. We present a study on open source software contribution in GitHub that focuses on the task of evaluating pull requests, which are one of the primary methods for contributing code in GitHub. We analyzed the association of various technical and social measures with the likelihood of contribution acceptance. We found that project managers made use of information signaling both good technical contribution practices for a pull request and the strength of the social connection between the submitter and project manager when evaluating pull requests. Pull requests with many comments were much less likely to be accepted, moderated by

Foundation [10], Apache Software Foundation [14], and Mozilla Foundation [21] officially describe themselves as meritocracies. For example, in the case of Mozilla, “authority is distributed to both volunteer and employed community members as they show their abilities through contributions to the project” [21]. These “abilities” are generally assumed to be technical expertise brought to the software project by various developers.

Previous studies on open source software suggest that there are many more factors that influence contribution evaluation beyond technical merit. In fact, prior work suggests that there exists a complex social structure around contribution in open source software [8]. New contributors to traditional open source projects are expected to “lurk” or monitor project mailing lists before even attempting contributions. These projects have complex socialization processes that need to be undertaken before accepting technical contributions [17].

With the advent of social media and distributed version control systems, many open source software projects operate with an

Quando os testes estão incluso, o PR  
tem **17%** mais chance de ser aceito

fator técnico

Tsay, Jason, Laura Dabbish, and James Herbsleb. "Influence of social and technical factors for evaluating contribution in GitHub."  
*Proceedings of the 36th international conference on Software engineering*. ACM, 2014.



Se a pessoa autora segue a pessoa responsável pelo projeto, tem **187%** mais chance do PR ser aceito

fator social

Tsay, Jason, Laura Dabbish, and James Herbsleb. "Influence of social and technical factors for evaluating contribution in GitHub." *Proceedings of the 36th international conference on Software engineering*. ACM, 2014.



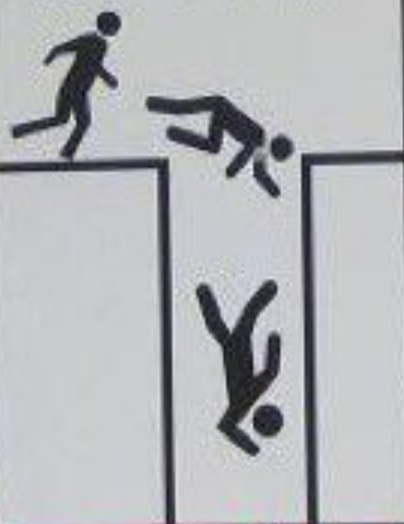
formalize as recomendações,  
crie guidelines sobre  
aspectos **comportamentais**



comunicação  
verbal, não verbal e escrita

# DANGER

DON'T RUN



BEWARE!



DEEP  
SHAFTS

DON'T WALK  
BACKWARDS



## UNMARKED HOLES

OLD TIMERS  
MINE

# comportamentos tóxicos



# COMPORTAMENTOS TÓXICOS

Impedem inovações e ideias

Promovem a cultura da não-comunicação

Colocam **o projeto e negócio em risco** por centralizar informação

Comunicação agressiva (verbal, não-verbal e escrita)

"como assim você não sabe isso???"



"como deixaram você entrar aqui??"

"vou ter que te explicar de novo?"

A análise de sentimento em comentários tem mostrado evidências de que comentários com **tom negativo** tendem a ser menos úteis

Sadowski, Caitlin, et al. "Modern code review: a case study at Google." *Proceedings of the 40th International Conference on Software Engineering: Software Engineering in Practice*. ACM, 2018

como evitar isso?



# Faça reviews como seres humanos

## How to Do Code Reviews Like a Human (Part One)

October 12, 2017

🕒 19 minute read

Lately, I've been reading articles about best practices for code reviews. I notice that these articles focus on finding bugs to the exclusion of almost every other component of a review. Communicating issues you discover in a constructive and professional way? Irrelevant! Just identify all the bugs, and the rest will take care of itself.

So I had a revelation: if this works for code, why not romance? With that, I'm announcing my new ebook to help developers with their love lives:

<https://mtlynch.io/human-code-reviews-1/>  
<https://mtlynch.io/human-code-reviews-2/>

ask, don't tell

ok, é só perguntar

ok, é só perguntar





"Testes não são importantes pra vc?"

pergunta sarcástica, com julgamento pessoal

"Testes não são importantes pra vc?"

pergunta sarcástica, com julgamento pessoal



"Esse PR não pode ser mergeado"

comentário opinativo, sem ação concreta, imperativo

"Esse PR não pode ser mergeado"

comentário opinativo, sem ação concreta, imperativo



"Por que não criou uma nova classe?"

pergunta com julgamento pessoal ainda  
"como você não pensou nisso?"

"Por que não criou uma nova classe?"

pergunta com julgamento pessoal ainda  
"como você não pensou nisso?"



busque comentar  
de maneira construtiva

"O que você acha sobre extrair essa lógica para uma classe? Acredito que vai melhorar a legibilidade e reduzir a complexidade"

construtivo





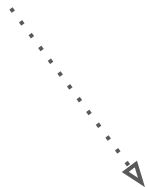
"Não sei se você já analisou isso,  
mas será que não vale a pena criar  
uma nova classe para esse caso?"

sem suposição, tom de sugestão

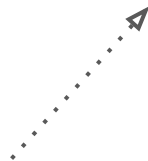




definição

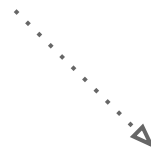


desafios



práticas do  
dia a dia

aprendizados



desenvolvimento de software tem  
muito a ver com **cultura**

"A cultura não faz as pessoas,  
as pessoas fazem a cultura"

Chimamanda Ngozi Adichie





olhe para o seu time



diversidade ajuda a estimular empatia

pode ajudar a reduzir  
comportamentos tóxicos

e impactar positivamente  
na inovação e lucro

## **Diversidade de gênero:**

21% mais chances de resultados  
acima da média do mercado

## **Diversidade cultural e étnica:**

33% mais chances de resultados  
acima da média do mercado

<https://assets.mckinsey.com/~media/857F440109AA4D13A54D9C496D86ED58.ashx>

olhe também para o ambiente

fatores não-técnicos

pressão, sobrecarga de atividades,  
experiência e contexto de negócio

Baysal, Olga, et al. "The influence of non-technical factors on code review." *2013 20th Working Conference on Reverse Engineering (WCRE)*. IEEE, 2013.

a qualidade do software  
reflete todos esses fatores

impacta também no código escrito



código escrito é uma  
forma de **comunicação**

"Instead of imagining that our main task is to instruct a computer what to do, let us concentrate rather on **explaining to human beings** what we want a computer to do."

**code review** é sobre cultura,  
pessoas, qualidade de software

e aí, como é o **code review**  
no seu dia a dia?

minhas referências

# Code Reviewing in the Trenches

## Challenges and Best Practices

Laura MacLeod and Michaela Greiler, Microsoft

Margaret-Anne Storey, University of Victoria

Christian Bird, Microsoft Research

Jacek Czerwonka, Microsoft

// *A large-scale study of Microsoft developers*

developers' code review practices to summarize the challenges that code-change authors and reviewers face, suggest best code-reviewing practices, and discuss tradeoffs that practitioners should consider.

### The Code Review Study

To understand code review processes, researchers generally focus on a retrospective analysis of code review trace data (for example, from CodeFlow,<sup>1</sup> GitHub pull requests,<sup>2</sup> and emails<sup>3</sup>). In addition, some researchers have conducted interviews or surveys<sup>2,4</sup> to reveal the motivations for and challenges of code review. For example, Alberto Bacchelli and Christian Bird interviewed developers while they performed code reviews.<sup>1</sup>

To gain a more in-depth understanding of the human and social factors that drive code review in a large industrial context, we investigated

## Modern Code Review: A Case Study at Google

Caitlin Sadowski, Emma Söderberg,  
Luke Church, Michal Sipko  
Google, Inc.

{supertri,emso,lukechurch,sipkom}@google.com

Alberto Bacchelli  
University of Zurich  
bacchelli@ifi.uzh.ch

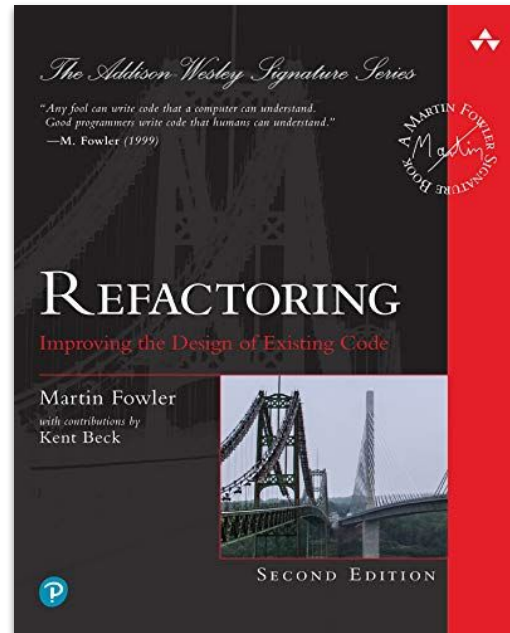
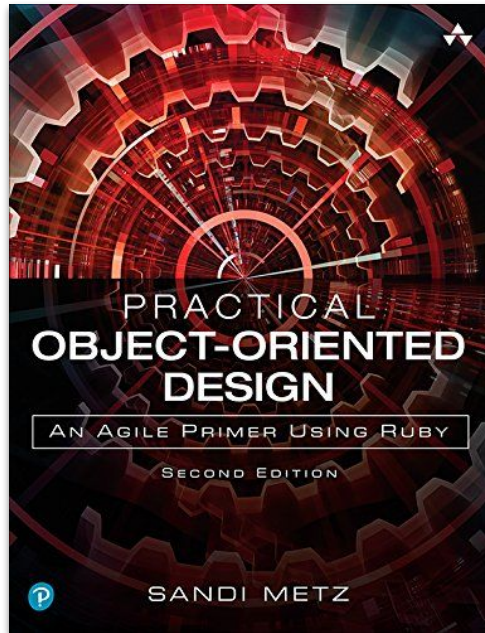
### ABSTRACT

Employing lightweight, tool-based code review of code changes (aka *modern code review*) has become the norm for a wide variety of open-source and industrial systems. In this paper, we make an exploratory investigation of modern code review at Google. Google introduced code review early on and evolved it over the years; our study sheds light on why Google introduced this practice and analyzes its current status, after the process has been refined through decades of code changes and millions of code reviews. By means of 12 interviews, a survey with 44 respondents, and the analysis of review logs for 9 million reviewed changes, we investigate motivations behind code review at Google, current practices, and developers' satisfaction and challenges.

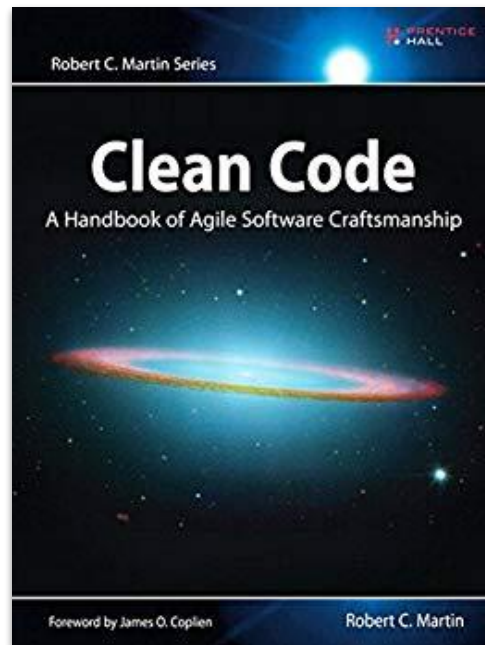
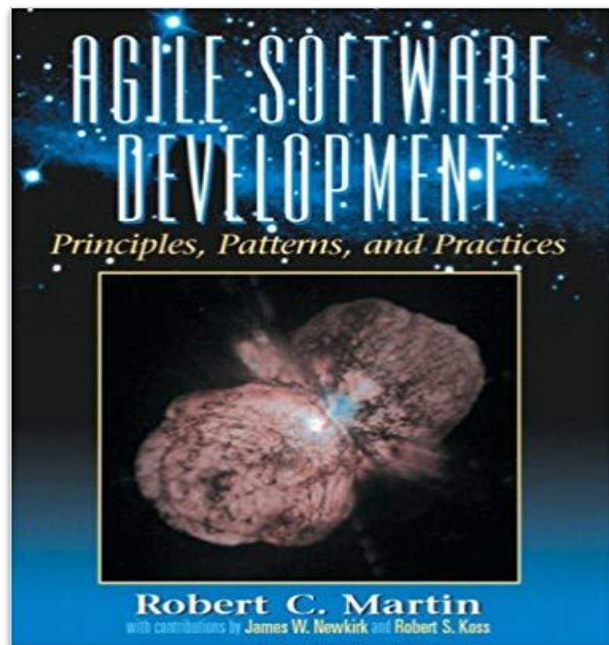
### CCS CONCEPTS

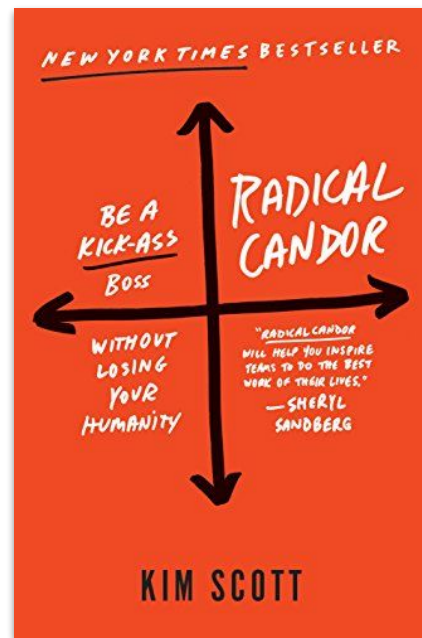
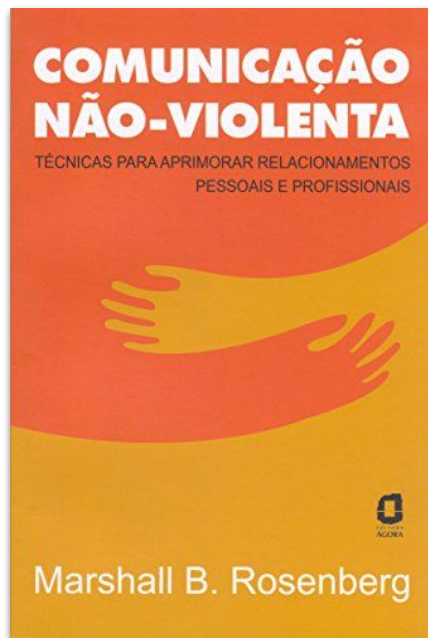
- **Software and its engineering** → **Software maintenance tools**;

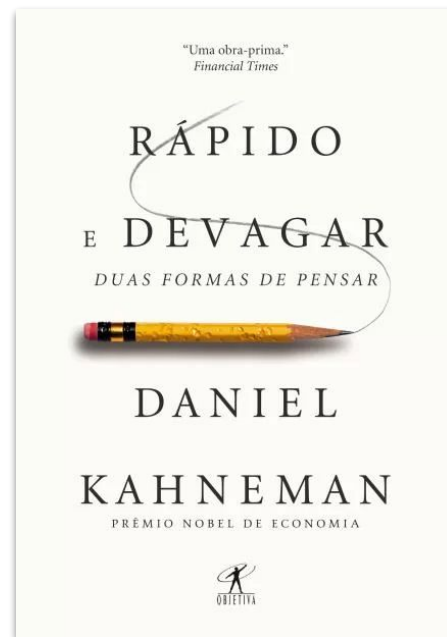
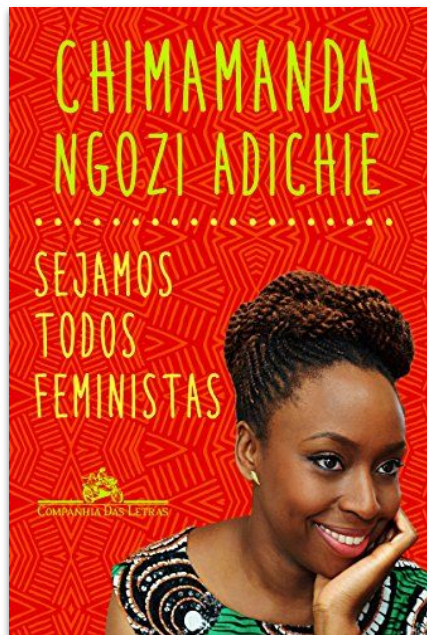
An open research challenge is understanding which practices represent valuable and effective methods of review in this novel context. Rigby and Bird quantitatively analyzed code review data from software projects spanning varying domains as well as organizations and found five strongly convergent aspects [33], which they conjectured can be prescriptive to other projects. The analysis of Rigby and Bird is based on the value of a *broad* perspective (that analyzes multiple projects from different contexts). For the development of an empirical body of knowledge, championed by Basili [7], it is essential to also consider a *focused and longitudinal* perspective that analyzes a single case. This paper expands on work by Rigby and Bird to focus on the review practices and characteristics established at Google, *i.e.*, a company with a multi-decade history of code review and a high-volume of daily reviews to learn from. This paper can be (1) prescriptive to practitioners performing code review and (2) compelling for researchers who want to understand and support this novel process.

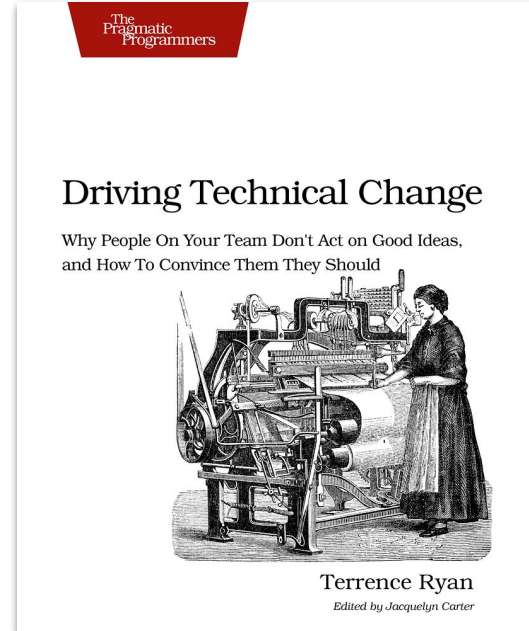
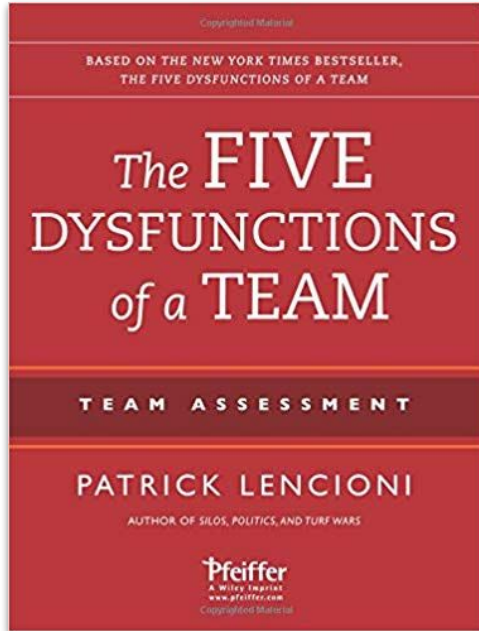












# Decoding the representation of code in the brain: An fMRI study of code review and expertise

Benjamin Floyd  
University of Virginia  
bef2cj@virginia.edu

Tyler Santander  
University of Virginia  
ts7ar@virginia.edu

Westley Weimer  
University of Virginia  
weimer@virginia.edu

**Abstract**—Subjective judgments in software engineering tasks are of critical importance but can be difficult to study with conventional means. Medical imaging techniques hold the promise of relating cognition to physical activities and brain structures. In a controlled experiment involving 29 participants, we examine code comprehension, code review and prose review using functional magnetic resonance imaging. We find that the neural representations of programming languages vs. natural languages are distinct. We can classify which task a participant is undertaking based solely on brain activity (balanced accuracy 79%,  $p < 0.001$ ). Further, we find that the same set of brain regions distinguish between code and prose (near-perfect correlation,  $r = 0.99$ ,  $p < 0.001$ ). Finally, we find that task distinctions are modulated by expertise, such that greater skill predicts a less differentiated neural representation ( $r = -0.44$ ,  $p = 0.016$ ) indicating that more skilled participants treat code and prose more similarly at a neural activation level.

**Keywords**—medical imaging; code comprehension; prose review

among both clinical and psychological researchers. Unlike other cognitive neuroscience methods (e.g., EEG or PET), fMRI allows for rapid sampling of neural signal across the whole brain (1–2 seconds) and offers high spatial resolution (scale of millimeters) with regard to localizing signal sources. Thus, fMRI arguably provides the best available measure of online neural activity in the living, working human brain.

We present an fMRI study of software engineering activities. We focus on understanding code review, its relationship to natural language, and expertise. We note that the use of fMRI in software engineering is still exploratory; to the best of our knowledge this is only the second paper to do so [70], and is the first to consider code review and expertise. We explore these tasks because developers spend more time understanding code than any other activity [18], [29], [59], [62]. A NASA survey, for example, ranked understanding as more important than functional correctness when making use of software [53]. Similarly, with companies such as Facebook [77] and Google [36] mandating code review for

- ✓ [guidelines.plataformatec.com.br](https://guidelines.plataformatec.com.br)
- ✓ [github.blog/2015-01-21-how-to-write-the-perfect-pull-request](https://github.blog/2015-01-21-how-to-write-the-perfect-pull-request)
- ✓ [medium.com/palantir/19e02780015f](https://medium.com/palantir/19e02780015f)
- ✓ [medium.com/@jgefroh/f7ea1494d4c0](https://medium.com/@jgefroh/f7ea1494d4c0)
- ✓ [forbes.com/sites/quora/2014/11/07/10-characteristics-of-a-bad-software-engineer](https://forbes.com/sites/quora/2014/11/07/10-characteristics-of-a-bad-software-engineer)
- ✓ [blog.plataformatec.com.br/2018/07/como-evitar-silos-de-conhecimento-na-sua-codebase-e-levar-seus-code-reviews-para-o-proximo-nivel/](https://blog.plataformatec.com.br/2018/07/como-evitar-silos-de-conhecimento-na-sua-codebase-e-levar-seus-code-reviews-para-o-proximo-nivel/)

- ✓ **Building an Iconic Company - Reed Hasting**  
[youtube.com/watch?v=BsXXIfqbnRk](https://youtube.com/watch?v=BsXXIfqbnRk)
- ✓ **A Arquitetura (Peculiar) do Stack Overflow - Roberta Arcoverde**  
[infoq.com/br/presentations/a-arquitetura-peculiar-do-stack-overflow](https://infoq.com/br/presentations/a-arquitetura-peculiar-do-stack-overflow)
- ✓ **Arquitetura, pragmatismo e simplicidade - Roberta Arcoverde**  
[docs.google.com/presentation/d/1DMpfVcXtALeCPwQwTM0Nz-YE1DBz7hCvPMf8q6O1ogl/preview](https://docs.google.com/presentation/d/1DMpfVcXtALeCPwQwTM0Nz-YE1DBz7hCvPMf8q6O1ogl/preview)
- ✓ **Talking with Tech Leads - Patrick Kua**  
[youtube.com/watch?v=dNE6aqkG7ss](https://youtube.com/watch?v=dNE6aqkG7ss)

- ✓ **Implementing a Strong Code-Review Culture - Derek Prior**  
[youtube.com/watch?v=PJjmw9TRB7s](https://youtube.com/watch?v=PJjmw9TRB7s)
- ✓ **Maintaining a big open source project: lessons learned - Leonardo Tegon**  
[youtube.com/watch?v=rnOcDH\\_sgxg](https://youtube.com/watch?v=rnOcDH_sgxg)
- ✓ **Integração Discreta: como melhorar a Integração Contínua e ainda ganhar em colaboração - George Guimarães**  
[infoq.com/br/presentations/integracao-discreta-como-melhorar](https://infoq.com/br/presentations/integracao-discreta-como-melhorar)





**Roberta Arcoverde**

@rla4



Num ambiente de entrega contínua, diante de erros bestas (typos, por ex), você/seu time seguem o processo de mudança com branch/PR/code review, ou pulam alguma/todas essas etapas? Vai direto pra master? Pula só o code review? Thoughts?

[Translate Tweet](#)

7:14 PM · Feb 19, 2019 · [Twitter Web Client](#)

<https://twitter.com/rla4/status/1097982806163185666>





Codar:  
**COISA DE**  
Mulher



# MUITO OBRIGADA

[speakerdeck.com/elainenaomi](https://speakerdeck.com/elainenaomi)



**CÓDIGO: ELAINE**

50% por 6 meses