Código Limpo:

Habilidades Práticas do Agile Software

•••

Prefácio, Introdução e Capítulo 1

Este material foi desenvolvido pelo grupo PET/ADS do IFSP São Carlos

Prefácio

James O. Complien

O biscoito da sorte dinamarquês

“Honestidade em pequenas coisas não é uma coisa pequena.”

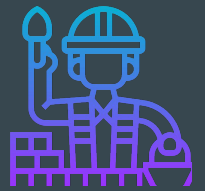
Dentro do contexto do livro:

1. Pequenas coisas são importantes.
2. O livro dá atenção às preocupações modestas cujos valores estão longe de ser pequenos.
3. Devemos ser honestos com o nosso código.

Manutenção: o ato de reparar

Na indústria de software 80% ou mais é “manutenção”

Produz-se pouco e se conserta muito



Ícones criados por Muhammad Ali e Eucalyp - Flaticon

5S: raízes de um profissionalismo responsável

SEIRI – Senso de Utilização

SEITON – Senso de Ordenação

SEISOU – Senso de Limpeza

SEIKETSU – Senso de Saúde

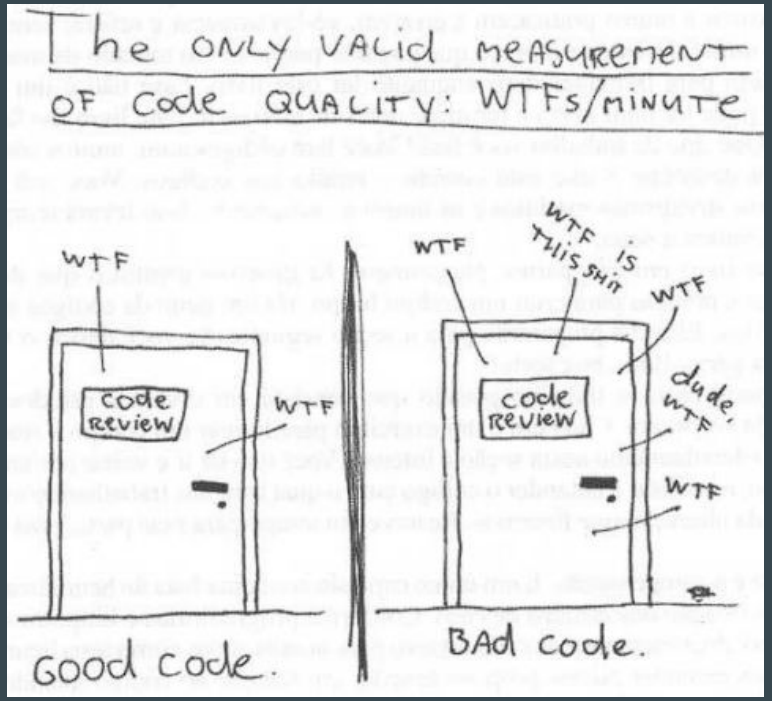
SHITSUKE – Senso de Autodisciplina

Atenção aos detalhes

- Além da metodologia japonesa “5S”, a prática de um bom software requer principalmente *“foco, presença de espírito e pensamento."*
- É necessário vivenciar o momento de escrita do código e dar atenção aos mínimos detalhes que ele exige.

Introdução

Introdução



Que porta representa seu código? Por que estamos nessa porta ou naquela?

A resposta:

Habilidade Profissional

Conhecimento

Adquire-se dos princípios, padrões, práticas e heurísticas.

Trabalho

É preciso esmiuçar esse conhecimento com o trabalho, com a prática. Deve-se errar, praticar sozinho e ver outras pessoas errarem.

O livro é dividido em três partes

- **Primeira:** descrição dos princípios, padrões e práticas de um código limpo.
- **Segunda:** estudos de caso e exercícios para limpar um código.
- **Terceira:** lista de heurísticas reunidas durante a criação dos estudos de caso.

Capítulo 1:
Código Limpo

O fim do código escrito está próximo?

Com certeza, **não**.

"Nunca nos livraremos dos códigos, pois eles representam os detalhes dos requisitos."

"O código é a linguagem na qual expressamos nossos requisitos."

Primeira Parte: Código Ruim

O código ruim é como se fosse um caminho penoso, com armadilhas ocultas e arbustos emaranhados. Não se sabe o que está acontecendo, só se vê um código cada vez mais sem sentido.

O exemplo do fracasso de uma empresa

Uma empresa dos anos 80 lançou um aplicativo muito bom que se tornou popular, mas ao longo de novas versões os bugs aumentaram muito. A empresa faliu.

“Foi o código ruim que acabou com a empresa.”

Um bom código importa: é a premissa mais robusta, apoiada e plena da nossa área. Isso é comprovado ao se lidar com a falta de um bom código durante muito tempo.

Mas por que isso aconteceu?

- Pressa
- Prazos curtos
- O cansaço de trabalhar por muito tempo em um código
- Querer terminar logo
- Enfim, o lema: uma bagunça que funciona é melhor do que nada

Só que não é bem assim...

Segunda Parte: O Custo de Ter um Código Confuso

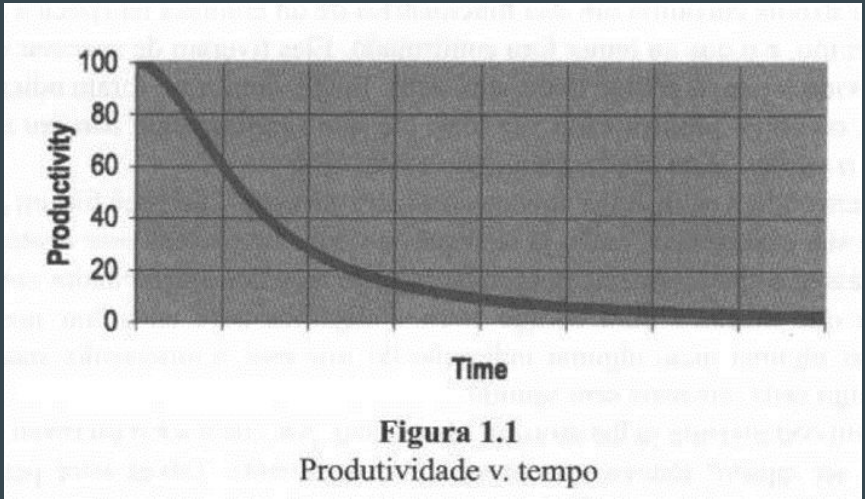


Figura 1.1
Produtividade v. tempo

Exemplo de uma equipe que trabalha rapidamente, mas percebe mais tarde que estão indo a passos de tartaruga

Produtividade zero

Cada alteração feita no código causa uma falha, que gera amarrações e confusões. Ao longo do tempo, não dá para arrumar mais nada.

Adicionar mais membros com a esperança de aumentar a produtividade pode aumentar a confusão.

Os novatos não conhecem o todo e não sabem necessariamente diferenciar uma mudança benéfica de uma ruim.

Conforme a bagunça aumenta, a produtividade da equipe diminui.

Terceira Parte: O Grande Replanejamento

No final, a equipe se rebela e exige um replanejamento do projeto. A gerência não tem escolha e acata essa decisão.

Um novo plano

Solução imediata:

Equipe Nova: começa do zero e realmente tenta fazer um trabalho belo

Equipe Antiga: deve continuar na manutenção do sistema atual

Mas os velhos problemas voltam...

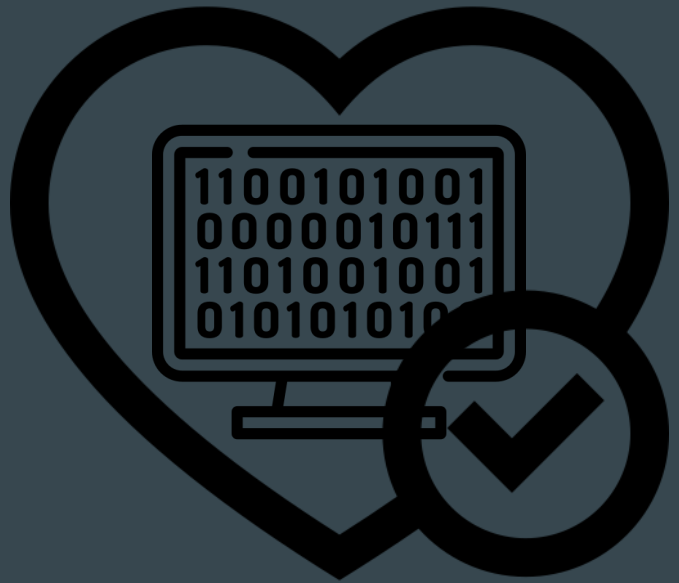
Os códigos são novos, mas as práticas são antigas. O novo código logo fica caótico novamente e tudo se repete.

Quarta Parte: Atitude

Temos grande responsabilidade sobre o código.

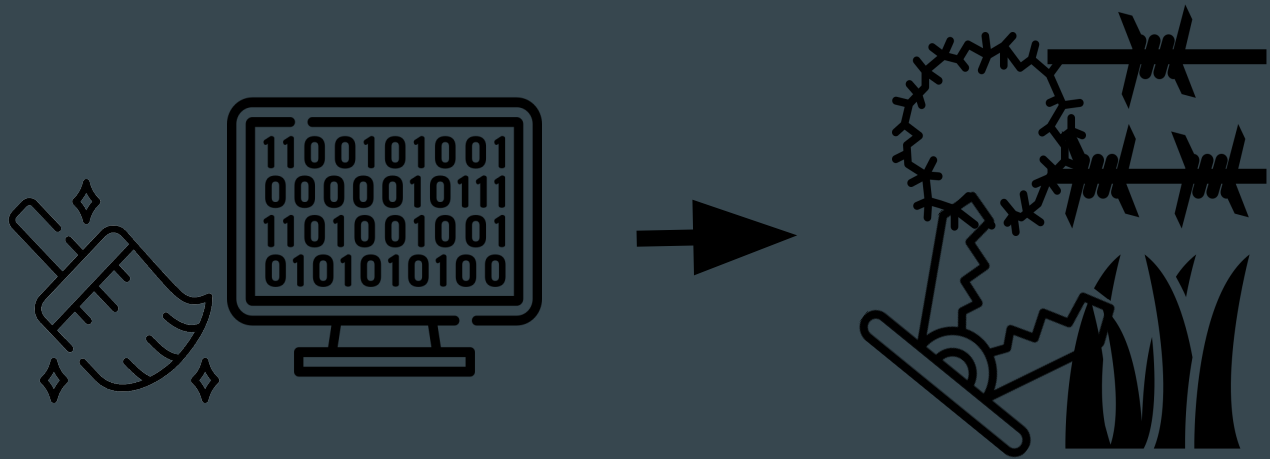
Os gerentes protegerão com paixão os prazos.

Os programadores devem proteger seu código com a mesma paixão, ao invés de simplesmente entregar uma bagunça.



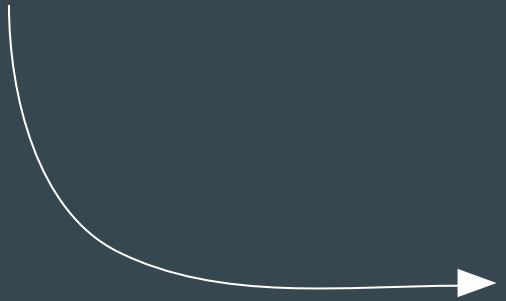
Quinta Parte: O Principal Dilema

A única maneira da bagunça não surgir, de prosseguir mais rápido e de não estragar todo um projeto é justamente a prática de manter um código limpo.



Sexta Parte: A Arte do Código Limpo?

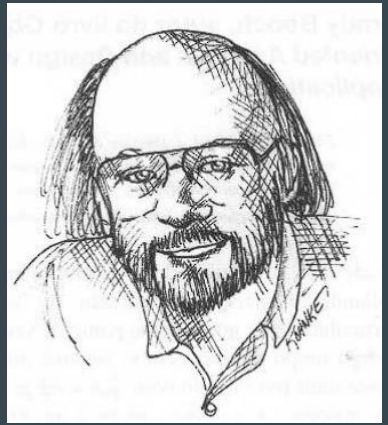
"Como escrevemos um código limpo?"



"Tendo sensibilidade ao código."

Código Limpo por Bjarne Stroustrup

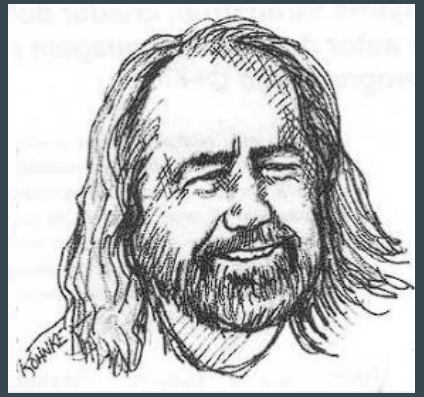
"Gosto do meu código elegante e eficiente. A lógica deve ser direta para dificultar o encobrimento de bugs, as dependências mínimas para facilitar a manutenção, o tratamento de erro completo de acordo com uma estratégia clara e o desempenho próximo do mais eficiente, de modo a não incitar as pessoas a tornarem o código confuso com otimizações sorrateiras. O código limpo faz bem apenas uma coisa."



Bjarne Stroustrup, criador do C++

Código Limpo por Grady Booch

“Um código limpo é simples e direto. Ele é tão bem legível quando uma prosa bem escrita. Ele jamais torna confuso o objetivo do desenvolvedor; em vez disso, ele está repleto de abstrações claras e linhas de controle objetivas.”



Grady Booch, autor do livro *Object Oriented Analysis and Design with Application*

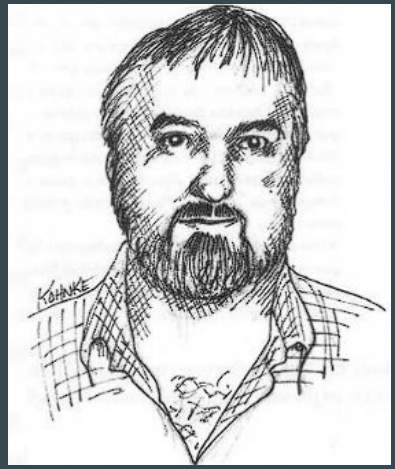
Código Limpo por Dave Thomas

"Além de seu criador, um desenvolvedor pode ler e melhorar um código limpo. Ele tem testes de unidade e de aceitação, nomes significativos; ele oferece apenas uma maneira, e não várias, de se fazer uma tarefa; possui poucas dependências, as quais são explicitamente declaradas e oferecem um API mínimo e claro. O código deve ser inteligível já que dependendo da linguagem, nem toda informação necessária pode ser expressa no código em si."

Dave Thomas, fundador da OTI,
o pai da estratégia Eclipse

Código Limpo por Michael Feathers

“Eu poderia listar todas as qualidades que vejo em um código limpo, mas há uma predominante que leva a todas as outras. Um código limpo sempre parece que foi escrito por alguém que se importava. Não há nada de óbvio no que se pode fazer para torná-lo melhor. Tudo foi pensado pelo autor do código, e se tentar pensar em algumas melhoras, você voltará ao início, ou seja, apreciando o código deixado para você por alguém que se importa bastante com essa tarefa.”



Michael Feathers, autor de *Working Effectively with Legacy Code*

Código Limpo por Ron Jeffries

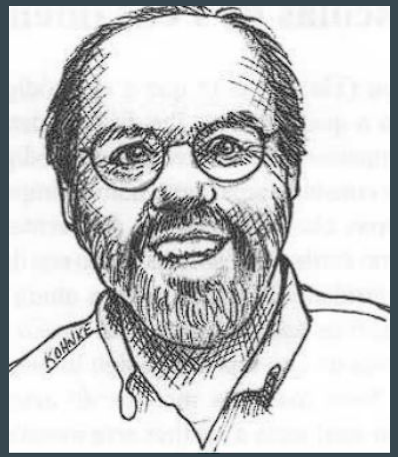
"Nestes anos recentes, comecei, e quase finalizei, com as regras de Beck sobre código simples. Em ordem de prioridade, são:

- *Efetue todos os testes;*
- *Sem duplicação de código;*
- *Expresse todas as ideias do projeto que estão no sistema;*
- *Minimize o número de entidades, como classes, métodos, funções e outras do tipo."*

Ron Jeffries, autor de *Extreme Programming Installed* e *Extreme Programming Adventures in C#*

Código Limpo por Ward Cunningham

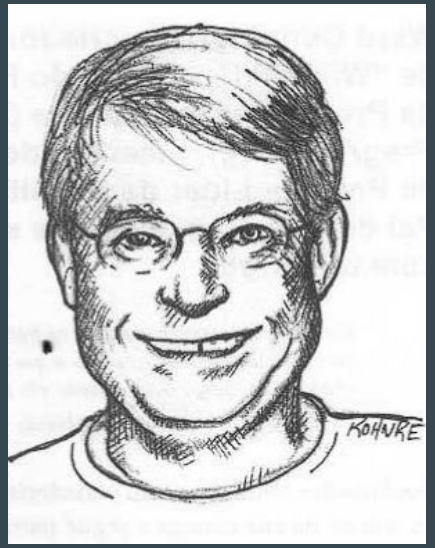
“Você sabe que está criando um código limpo quando cada rotina que você lê se mostra como o que você esperava. Você pode chamar de código belo quando ele também faz parecer que a linguagem foi feita para o problema.”



Ward Cunningham, líder da Smalltalk e da OO. Pai de todos aqueles que se importam com o código.

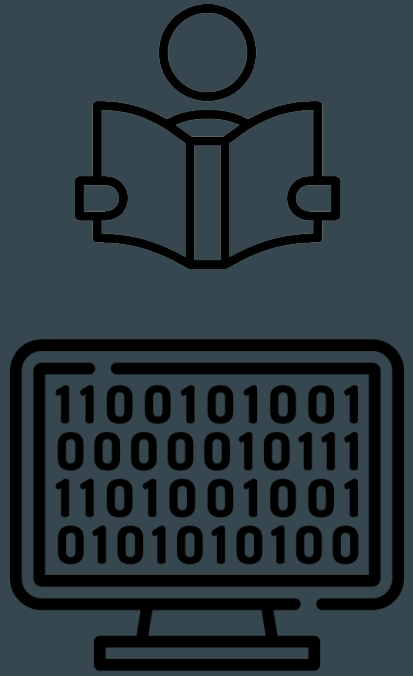
Escolas de Pensamento

E para Robert, o autor do livro?



Sétima Parte: Somos autores

"O quanto realmente se lê de um código?"

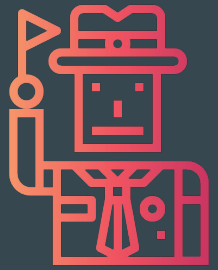


Você é um autor

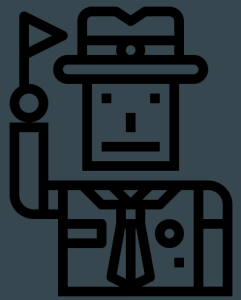
A maioria do trabalho não é escrever o código? Na verdade, não.

Constantemente lemos nosso código antigo enquanto estamos criando o novo. Quando você está programando, frequentemente você retoma e lê outras partes do seu código. Portanto, se quisermos que o código seja eficiente e de fácil escrita, devemos torná-lo de fácil leitura.

Nona Parte: A Regra do Escoteiro



“Deixe a área do acampamento mais limpa do que como você a encontrou.”



Reescrevendo aos poucos

Um bom código precisa ser mantido sempre limpo, mas não é necessário que se faça tudo de uma vez. Aos poucos, pode-se fazer:

- Trocar o nome de uma variável por uma melhor
- Dividir uma função que esteja muito grande
- Reduzir uma instrução do if aninhada

Conclusão: Código Limpo

- Código simples e direto
- Fácil de entender, cuja leitura seja natural
- Objetivos bem propostos e centralizados
- Deve conter apenas o necessário
- Código pequeno, na medida do possível
- Precisa de cuidado
- Atenção aos detalhes
- Testes e mais testes

Sobre o autor

Robert C. “Uncle Bob” Martin é desenvolvedor e consultor de software desde 1990. Ele é o fundador e o presidente da Object Mentor, Inc., uma equipe de consultores experientes que orientam seus clientes no mundo todo em C++, Java, C#, Ruby, OO, Padrões de Projeto, UML, Metodologias Agile e eXtreme Programming.

Este conjunto de slides foi elaborado a partir da obra:

MARTIN, Robert. **Código Limpo:**
Habilidades Práticas do Agile Software. 1. ed.
Rio de Janeiro: Alta Books, 2009.

A equipe

Bianca Antonietti, Autora

Aluna e bolsista do PET/ADS desde novembro de 2022
LinkedIn

Lucas Oliveira, Revisor

Professor de Computação, é tutor do PET/ADS desde janeiro de 2023.
LinkedIn