

# Qualidade de Software

UC: Gestão e Qualidade de Software

**Prof. Eliane Faveron Maciel**

UNIFACS  
ecossistema ânima

22 de março de 2024

# Overview

## 1. Qualidade de Software

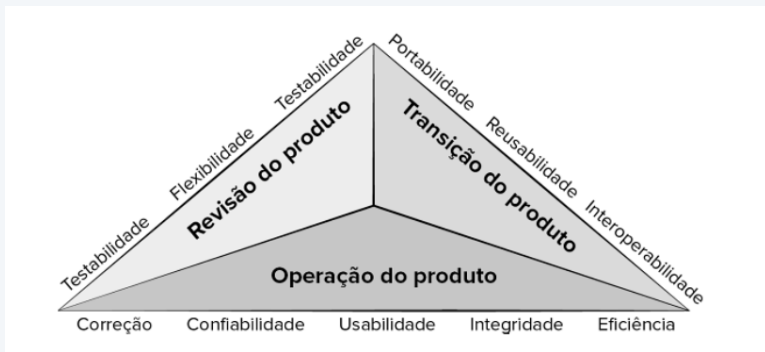
## 2. Referências

# Qualidade de Software

---

# O que é qualidade?

pode ser definida como: Uma gestão de qualidade efetiva aplicada de modo a criar um produto útil que forneça valor mensurável para aqueles que o produzem e para aqueles que o utilizam.



# O que impacta a qualidade de software?

O objetivo do controle da qualidade de software e da gestão da qualidade em geral é, em sentido mais amplo, eliminar problemas de qualidade no software. Tais problemas são conhecidos por diversos nomes – bugs, falhas, erros ou defeitos, apenas para citar alguns.

# Revisões

O objetivo de uma revisão técnica formal (RTF) é encontrar erros antes que eles ocorram.

1. descobrir erros na função, na lógica ou na implementação, para qualquer representação do software;
2. verificar se o software sob revisão satisfaz seus requisitos;
3. garantir que o software tenha sido representado de acordo com padrões predefinidos;
4. conseguir software que seja desenvolvido de modo uniforme;
5. tornar os projetos mais administráveis.

# O papel de revisor

”Todo código desenvolvido por um desenvolvedor tem que ser, em seguida, analisado por pelo menos um outro desenvolvedor, chamado de **revisor**. O **revisor** pode adicionar comentários no código sob revisão, procurando esclarecer dúvidas, sugerindo melhorias, indicando bugs, etc.”

1. Encontro de defeitos – para achar bugs;
2. Melhoria de código – para melhorar a consistência do código, legibilidade, etc.;
3. Soluções alternativas – para encontrar uma melhor implementação;
4. Transferência de conhecimento – para fins de aprendizagem;
5. Conscientização e transparência da equipe – para tornar a equipe ciente da evolução do código e tornar as alterações do código transparentes.
6. Compartilhamento de posse do código – relacionado a “Conscientização e Transparência da Equipe”, mas com conotação de colaboração.

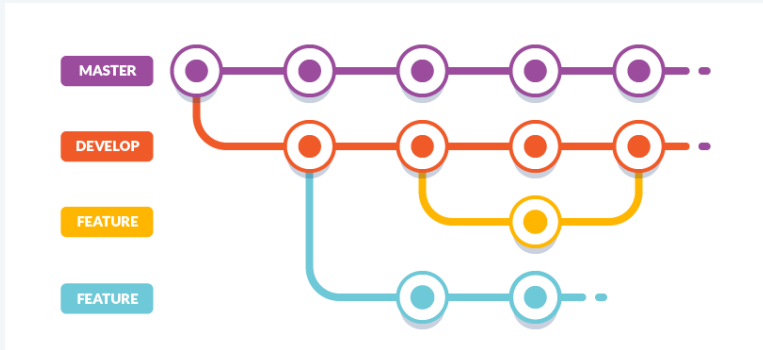
# Git e GitFlow

Empresas utilizam ferramentas de versionamento de código como GitHub, GitLab, BitBucket, essas por sua vez possibilitam que o código seja revisado antes que seja incorporado para o core do sistema.

1. Branch (ramo) - significa que você diverge da linha principal de desenvolvimento e continua a trabalhar sem alterar essa linha principal.
2. Commits - registra alterações em um ou mais arquivos no seu branch
3. Pull - é usado para buscar e baixar conteúdo de repositórios remotos e fazer a atualização imediata ao repositório local para que os conteúdos sejam iguais.
4. Push - é usado para gravar em um repositório remoto.
5. Merge - é o jeito do Git de unificar um histórico bifurcado



# GitFlow




# Pull Requests


É uma proposta para mesclar as alterações de um branch em outro. Em uma pull request, os colaboradores podem revisar e discutir o conjunto de alterações proposto antes de integrá-las à base de código principal.

estacionamento.java Outdated


```
2 +  
3 + public class Estacionamento {  
4 +  
5 +     public Hashtable<String, String> veiculos;
```

 **mtov** 5 days ago 😊 ...


Sugestão: "ocultar" essa tabela hash dos clientes da classe, tornando-a privada. E, também, criar um método "estaciona" que deve ser chamado toda vez que um novo cliente quiser estacionar um veículo.

 **alinedtorres** 3 days ago Author Owner 😊 ...

Agradeço a sugestão! Acabei de realizar as alterações sugeridas.

 **mtov** now 😊 ...

Obrigado, LGTM!



# Recomendações aos revisores

1. Revisores sempre devem lembrar que o objetivo da revisão é detectar problemas inequívocos no código submetido.
2. Evite comentários subjetivos e relacionados a estilos pessoais.
3. Nunca use palavras ofensivas, sarcásticas ou mesmo irônicas.
4. Sempre restrinja seus comentários ao código que foi submetido e evite tratar de assuntos pessoais.

# Recomendações aos revisores

1. Procure fazer perguntas e não julgamentos.
2. Se você tiver feito um comentário errado ou sem sentido.
3. Sempre que possível, use emojis, pois eles deixam a linguagem mais coloquial e amigável.
4. Sempre que for esclarecedor, referencie a documentação interna ou externa ao projeto. Isso vai ajudar a embasar seus comentários.
5. Não deixe de elogiar o código submetido.
6. Se necessário, use imagens e screenshots para explicar sua dúvida.
7. Sempre que for razoável, use o pronome nós ou a expressão a gente, em vez de usar o pronome você.
8. Se você tiver uma divergência muito forte em relação ao código submetido, tente agendar uma reunião com o autor para expor sua visão e tentarem chegar a um consenso.

## Recomendações aos autores

1. Revisão de código não é um exame de proficiência. Ou seja, como autor, não leve a revisão para o lado pessoal e nunca imagine que o revisor está julgando sua competência.
2. Submeta PRs pequenos, caso queiram obter uma resposta rápida e mais proveitosa dos revisores. Por exemplo, os autores do livro *Software Engineering at Google*, recomendam que um PR deve ter no máximo 200 linhas de código.
3. Se PRs forem muito grandes, existe também a chance de a qualidade da revisão cair muito.

# Automatizando a review

1. CodeFactor
2. Codacy
3. CodeClimate
4. SonarLint
5. Code Rabbit

# Referências

---

# Referências



Pressman, Roger (2021)

Engenharia de Software: Uma abordagem Profissional

*AMGH Editora Ltda – 9. ed.*



Sommerville, Ian (2011)

Engenharia de Software

*Pearson Prentice Hall – 9. ed.*



Marco Tulio Valente (2020)

Engenharia de Software Moderna: Princípios e Práticas para Desenvolvimento de Software com Produtividade

*Editora: Independente*





# Obrigada

**Prof. Eliane Faveron Maciel**

UNIFACS  
ecossistema ânima

22 de março de 2024