Requisitos de *Software*

Prof.ª Ma. Jessica Oliveira

Católica
Universidade Católica de Brasília

Aula 05 – 08/04/2025

Documentação de requisitos: especificação e padrões.

Universidade Católica de Brasília

Introdução.

- A documentação de requisitos é uma das etapas **mais críticas** no processo de desenvolvimento de software.
- Ela consiste no **registro estruturado e claro das necessidades, expectativas, restrições e funcionalidades que um sistema deverá atender**.
- Esses documentos servem como base para diversas atividades subsequentes, como o projeto de *software*, a implementação, os testes, a validação e a manutenção.

Papel no ciclo de vida do *software*.

- Durante o ciclo de vida de um *software*, a documentação de requisitos atua como elo de comunicação entre as partes interessadas.
- Clientes, desenvolvedores, gerentes de projeto, analistas de testes e mantenedores utilizam a documentação para garantir um entendimento comum e alinhado sobre o que o sistema deverá fazer.
- Ela oferece uma representação tangível dos acordos firmados entre os *stakeholders*.

Papel no ciclo de vida do *software*.

- Sem documentação adequada, os requisitos permanecem apenas na esfera da comunicação verbal, o que aumenta significativamente os riscos de:
  - **Interpretações divergentes** entre cliente e equipe de desenvolvimento;
  - **Omissão de funcionalidades** importantes;
  - **Erros de implementação** que impactam negativamente a qualidade do produto.

Contribuições da documentação.

- Comunicação formal entre as partes envolvidas;
- Rastreabilidade de decisões e alterações ao longo do projeto;
- Base para planejamento, orçamento e cronograma;
- Fundamento para os testes de validação e verificação;
- Redução de retrabalho e de custos relacionados a erros tardios.

Riscos da má documentação.

- Quando a documentação de requisitos é negligenciada, os impactos negativos se manifestam rapidamente. Entre os problemas mais comuns estão:
  - Especificações ambíguas ou incompletas;
  - Falta de versionamento ou atualização de documentos;
  - Requisitos conflitantes;
  - Redução da qualidade final do software;
  - Aumento do custo de manutenção e evolução do sistema.
- Portanto, a documentação não deve ser vista como um processo burocrático, mas como um recurso estratégico para o sucesso do projeto.

Padrão IEEE 830.

Católica
Universidade Católica de Brasília

Visão geral.

- O padrão IEEE 830 é uma norma estabelecida pelo *Institute of Electrical and Electronics Engineers* (IEEE) para especificar os requisitos de software de forma estruturada, precisa e compreensível.
- Ele é voltado à elaboração de um documento conhecido como SRS – *Software Requirements Specification* (Especificação de Requisitos de Software).
- O IEEE 830 visa fornecer um modelo padronizado de documentação, que possa ser utilizado em diferentes projetos e domínios de *software*, promovendo clareza, consistência e completude nos requisitos.

Estrutura do documento SRS (IEEE 830).

Introdução.

- Essa seção tem como objetivo contextualizar o sistema que será desenvolvido. Inclui:
  - **Propósito** do documento e do sistema;
  - **Escopo**: uma visão geral do que o sistema irá fazer, destacando seus objetivos e benefícios;
  - **Definições**, **acrônimos** e **abreviações**;
  - **Referências** utilizadas para fundamentar o documento;
  - **Visão geral** da organização do documento.

Descrição geral.

- Essa seção descreve aspectos amplos do sistema, de forma mais conceitual, incluindo:
  - **Perspectiva do produto**: como o sistema se encaixa no ambiente existente (sistema autônomo, parte de um sistema maior, integração com outros sistemas);
  - **Funções do produto**: visão geral das funcionalidades sem entrar em detalhes de cada uma;
  - **Características dos usuários**: conhecimento técnico, necessidades, limitações;
  - **Restrições**: tecnológicas, legais, regulatórias ou outras;
  - **Suposições** e **dependências**.

Requisitos específicos.

- Essa seção detalha cada funcionalidade e requisito esperado, incluindo:
  - **Requisitos funcionais:** descrições das funções específicas que o sistema deve executar. Cada requisito deve conter um identificador único, descrição detalhada, entradas, saídas e resposta a condições anômalas.
  - **Requisitos não funcionais:** relacionados a desempenho, segurança, disponibilidade, usabilidade, entre outros.
  - **Requisitos de interface externa:** interfaces com outros sistemas, com hardware ou com o usuário.
  - **Requisitos de desempenho e qualidade:** tempo de resposta, capacidade de armazenamento, tolerância a falhas.

Vantagens e desvantagens do IEEE 830.

Pontos fortes.

- Proporciona padronização e organização das informações;
- Permite rastreabilidade de requisitos;
- Facilita o entendimento técnico por diferentes perfis profissionais;
- Serve como base para contrato entre cliente e equipe de desenvolvimento.

Pontos fracos.

- Pode gerar documentos extensos e difíceis de manter;
- Requer um esforço inicial considerável para elaboração;
- Em contextos ágeis ou altamente dinâmicos, pode ser inflexível;
- Muitas vezes é produzido e não mantido atualizado, tornando-se obsoleto.

Práticas de documentação em metodologias ágeis.

Católica
Universidade Católica de Brasília

Importante saber...

- Em contraste com abordagens tradicionais como o IEEE 830, as metodologias ágeis enfatizam a documentação leve e contínua.
- Segundo o Manifesto Ágil, "*software* em funcionamento é mais importante que documentação abrangente", o que **não significa a eliminação da documentação**, mas sim o foco no essencial e útil.

Documentação essencial no ágil.

- A documentação nas metodologias ágeis deve ser:
  - **Enxuta:** conter apenas o necessário para guiar o desenvolvimento;
  - **Colaborativa:** construída com o envolvimento do time e *stakeholders*;
  - **Atualizada frequentemente:** alinhada ao ritmo das iterações (*sprints*);
  - **Focada em valor:** refletir o que realmente é entregue ao cliente.

Product Backlog.

- É uma lista priorizada e evolutiva de funcionalidades, melhorias, correções e tarefas necessárias para o produto.
- Ele é mantido pelo *Product Owner* e contém histórias de usuário, que representam o ponto de vista do cliente/usuário.
- Essas histórias podem incluir critérios de aceite (*Conditions of Satisfaction*), que orientam o time sobre como validar que a funcionalidade está completa.

Definition of Done (DoD).

- É um acordo entre os membros da equipe sobre o que significa que um item do *backlog* está realmente concluído.
- Ele atua como um *checklist* de qualidade e inclui critérios técnicos e funcionais.
- Exemplo de DoD para uma *User Story*:
  - Código implementado e testado;
  - Testes unitários criados com 90% de cobertura;
  - Documentação técnica atualizada;
  - Funcionalidade validada pelo Product Owner.
- O DoD traz clareza, padronização e transparência à entrega de valor.

Vantagens e limitações da documentação ágil.

Pontos fortes.

- Foco na entrega de valor ao cliente;
- Flexibilidade para lidar com mudanças frequentes;
- Redução de desperdícios com documentação desnecessária;
- Incentiva o diálogo direto entre stakeholders e equipe técnica.

Pontos fracos.

- Riscos de perda de conhecimento em projetos de longo prazo;
- Documentação muitas vezes espalhada entre ferramentas (Jira, Trello, Git, Confluence);
- Pode ser incompleta ou desorganizada sem disciplina;
- Exige maturidade da equipe para manter consistência e rastreabilidade.

Critério	IEEE 830 (Tradicional)	Ágil (Scrum, XP, etc.)
Tipo de documentação	Formal e extensa	Enxuta e contínua
Foco	Registro detalhado e completo	Comunicação direta e entrega funcional
Forma	Documento técnico estruturado	User Stories e Backlogs
Atualização	Manual e periódica	Frequente e integrada ao processo
Adequação	Projetos grandes e contratos formais	Equipes pequenas, projetos dinâmicos
Vantagem	Clareza, rastreabilidade, base contratual	Flexibilidade, agilidade, foco no cliente
Desvantagem	Burocracia, risco de obsolescência	Risco de falta de histórico e inconsistência

Vamos para um desafio?

Católica
Universidade Católica de Brasília

Dúvidas?

jessica.oliveira@p.ucb.br