

Engenharia de Requisitos

UC: Modelos, Métodos e Técnicas de Engenharia de Software

Prof. Eliane Faveron Maciel

UNIFACS
ecossistema ânima

25 de março de 2024

Tópicos da Disciplina

- **Tópico 01** – Introdução de Engenharia Software
- **Tópico 02** – Modelos de Processo de Desenvolvimento
- **Tópico 03** – Engenharia de Requisitos
- **Tópico 04** – Visão e análise de projeto
- **Tópico 05** – Paradigma de desenvolvimento ágil
- **Tópico 06** – DevOps

Overview

1. Elicitação de Requisitos
2. Processo de Elicitação de requisitos
3. Requisitos no Desenvolvimento Ágil
4. Casos de uso
5. Referências

... última aula ...

- Engenharia de Requisitos
- Requisitos Funcionais
- Requisitos Não Funcionais

Elicitação de Requisitos

Documento de Especificação de Requisitos

- Requisitos devem estar corretos.
- Requisitos devem ser precisos, isto é, não devem ser ambíguos.
- Requisitos devem ser completos.
- Requisitos devem ser consistentes.
- Requisitos devem ser verificáveis, isto é, deve ser possível testar se os requisitos estão sendo atendidos.

Padrão IEEE 830

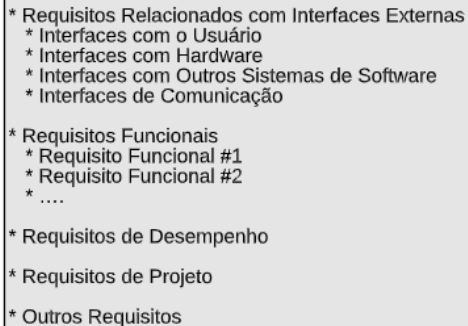
- 
- The diagram illustrates the structure of a Requirements Document according to the IEEE 830 standard. It is presented as a list of categories, each preceded by an asterisk. The categories are: Requirements Related to External Interfaces, Functional Requirements, Performance Requirements, Project Requirements, and Other Requirements. The 'Requirements Related to External Interfaces' category is further detailed with sub-items: Interfaces with the User, Interfaces with Hardware, Interfaces with Other Software Systems, and Communication Interfaces. The 'Functional Requirements' category is detailed with 'Functional Requirement #1', 'Functional Requirement #2', and an ellipsis indicating further requirements.
- * Requisitos Relacionados com Interfaces Externas
 - * Interfaces com o Usuário
 - * Interfaces com Hardware
 - * Interfaces com Outros Sistemas de Software
 - * Interfaces de Comunicação
 - * Requisitos Funcionais
 - * Requisito Funcional #1
 - * Requisito Funcional #2
 - *
 - * Requisitos de Desempenho
 - * Requisitos de Projeto
 - * Outros Requisitos

Figura: Documento de Requisitos no Padrão IEEE 830 [Valente, 2020].

Especificações estruturadas

No livro de [Sommerville, 2011] especifica que os requisitos devem ser escritos em um formato padrão. Por exemplo, podem ser abordados modelos de templates para representar com formulários estruturados.

Quadro 4.2

Exemplo de requisitos para o sistema de software de bomba de insulina.

3.2 O sistema deve medir o açúcar no sangue e fornecer insulina, se necessário, a cada dez minutos. *(Mudanças de açúcar no sangue são relativamente lentas, portanto, medições mais frequentes são desnecessárias; medições menos frequentes podem levar a níveis de açúcar desnecessariamente elevados.)*

3.6 O sistema deve, a cada minuto, executar uma rotina de autoteste com as condições a serem testadas e as ações associadas definidas na Quadro 4.3 *(A rotina de autoteste pode descobrir problemas de hardware e software e pode alertar o usuário para a impossibilidade de operar normalmente.)*

Figura: [Sommerville, 2011] - Engenharia de Software (cap.4-pag.67)

Quadro 4.3

Uma especificação estruturada de um requisito para uma bomba de insulina.

Bomba de insulina/Software de controle/SRS/3.3.2

Função	Calcula doses de insulina: nível seguro de açúcar.
Descrição	Calcula a dose de insulina a ser fornecida quando o nível de açúcar está na zona de segurança entre três e sete unidades.
Entradas	Leitura atual de açúcar (r_2), duas leituras anteriores (r_0 e r_1).
Fonte	Leitura atual da taxa de açúcar pelo sensor. Outras leituras da memória.
Saídas	CompDose — a dose de insulina a ser fornecida.
Destino	Loop principal de controle.
Ação	CompDose é zero se o nível de açúcar está estável ou em queda ou se o nível está aumentando, mas a taxa de aumento está diminuindo. Se o nível está aumentando e a taxa de aumento está aumentando, então CompDose é calculado dividindo-se a diferença entre o nível atual de açúcar e o nível anterior por quatro e arredondando-se o resultado. Se o resultado é arredondado para zero, então CompDose é definida como a dose mínima que pode ser fornecida.
Requisitos	Duas leituras anteriores, de modo que a taxa de variação do nível de açúcar pode ser calculada.
Pré-condição	O reservatório de insulina contém, no mínimo, o máximo de dose única permitida de insulina.
Pós-condições	r_0 é substituída por r_1 e r_1 é substituída por r_2 .
Efeitos colaterais	Nenhum.

Figura: [Sommerville, 2011] – Engenharia de Software (cap.4–pag.67)

Formulário de especificação de RF

1. A descrição da função ou entidade a ser especificada.
2. Uma descrição de suas entradas e de onde elas vieram.
3. Uma descrição de suas saídas e para onde elas irão.
4. Informações sobre a informação necessária para o processamento ou outras entidades usadas no sistema (a parte 'requires').
5. Uma descrição da ação a ser tomada.
6. Se uma abordagem funcional é usada, uma pré-condição define o que deve ser verdade antes que a função seja chamada, e é chamada uma pós-condição, especificando o que é verdade depois da função.
7. Uma descrição dos efeitos colaterais da operação (caso haja algum).

Processo de Elicitação de requisitos

Elicitação e análise de requisitos



Figura: [Sommerville, 2011] – Engenharia de Software (cap.4–pag.67)

Requisitos no Desenvolvimento Ágil

Histórias de Usuários

Segundo o autor [Valente, 2020], utilizar documentos complexos de elicitação de requisitos como vistos anteriormente podem levar muito tempo sendo escritos e ficar obsoletos ao longo do desenvolvimento.

A metodologia ágil trouxe uma nova forma de escrever esses requisitos chamada de **Histórias de Usuários**.

“Uma história de usuário descreve uma funcionalidade que tem valor tanto para o usuário final quanto para o cliente/comprador do projeto” (Mike Cohn)

Histórias de Usuários

A história de usuário é composta de três partes que começam com a letra C.

Historia de Usuario = Carto + Conversas + Confirmao

- **Cartão**, usado pelos clientes para escrever, na sua linguagem e em poucas sentenças, uma funcionalidade que esperam ver implementada no sistema.
- **Conversas** entre clientes e desenvolvedores, por meio das quais os clientes explicam e detalham o que escreveram em cada cartão.
- **Confirmação**, que é basicamente um teste de alto nível – especificado pelo cliente – para verificar se a história foi implementada conforme esperado. Exemplos e casos de teste que o cliente irá usar para confirmar a implementação da história. Por isso, são também chamados de testes de aceitação de histórias.

Histórias de Usuários

COMO UM [papel de usuário]
EU QUERO [realizar algo com o sistema]
PARA [motivo ou resultado]

Tabela: Modelo de Cartão

- Histórias devem ser **independentes**
- Histórias devem ser abertas para **negociação**.
- Histórias devem agregar **valor** para o negócio dos clientes.
- Deve ser viável **estimar** o tamanho de uma história.
- Histórias devem ser **sucintas** e pequenas.
- Histórias devem ser **testáveis**, isto é, elas devem ter critérios de aceitação objetivos.

Critérios de Aceitação:

São uma lista de verificação se a História foi implementada de acordo.

- Definir limites para a História.
- Ajudar o Dono do Produto detalhar em alto nível o que é necessário para entregar valor ao cliente.
- Ajudar o Time a entender melhor o objetivo da história.
- Informar ao Time de Desenvolvimento quando parar de adicionar funcionalidades à história.

Critério #1: exibir as últimas compras realizadas entre um período informado
Critério #2: exibir consulta apenas dos clientes adimplentes

Tabela: Modelo: Critérios de Aceitação

Testes de Aceitação

São testes criados a partir de aplicações de exemplos aos Critérios de Aceite.

Critério de Aceitação

Critério #1: somente podemos aceitar cartões de crédito com bandeiras com que temos convênio

Testes de Aceitação

- Comprador de Livros utiliza cartão de crédito Visa: Aceitou = correto. – Recusou = errado, deve ser corrigido!
- Comprador de Livros utiliza cartão de crédito MasterCard: – Aceitou = correto. – Recusou = errado, deve ser corrigido!

Testes de Aceitação

Dado	Descreve um contexto inicial no sistema.
Quando	Define um evento ou uma ação.
Então	Descreve um resultado esperado.

Tabela: Estrutura: Dado ... Quando ... Então ...

Casos de uso

“

Transferir Valores entre Contas

Ator: Cliente do Banco

Fluxo normal:

1 - Autenticar Cliente

2 - Cliente informa agência e conta de destino da transferência

3 - Cliente informa valor que deseja transferir

4 - Cliente informa a data em que pretende realizar a operação

5 - Sistema efetua transferência

6 - Sistema pergunta se o cliente deseja realizar uma nova transferência

Extensões:

2a - Se conta e agência incorretas, solicitar nova conta e agência

3a - Se valor acima do saldo atual, solicitar novo valor

4a - Data informada deve ser a data atual ou no máximo um ano a frente

5a - Se data informada é a data atual, transferir imediatamente

5b - Se data informada é uma data futura, agendar transferência

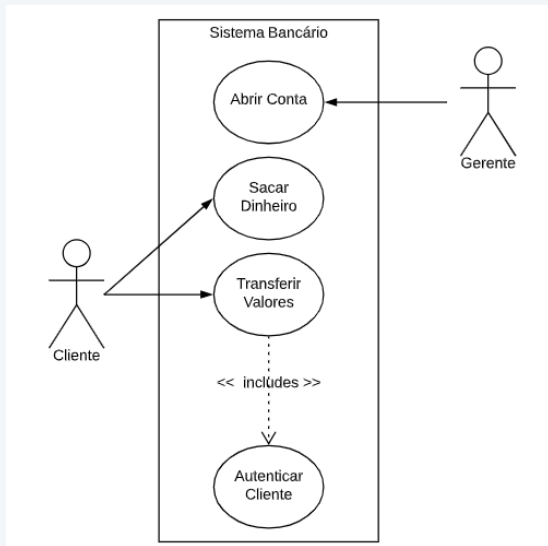


Figura: [Valente, 2020] - Exemplo de Diagrama UML de Casos de Uso (cap.3)

Mãos na massa ...

- Crie os cartões com os requisitos para o software que será desenvolvido na A3.
- Todos os cartões devem conter o critérios de aceite e testes de aceitação.
- Crie os cenários de **casos de uso**. Podem escolher se escreverão em formato de Diagrama UML ou documentos textuais.

Referências

Referências



Sommerville, Ian (2011)

Engenharia de Software

Pearson Prentice Hall – 9. ed.



Pressman, Roger (2021)

Engenharia de Software: Uma abordagem Profissional

AMGH Editora Ltda – 9. ed.



Valente, Marco Tulio (2020)

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

Editora: Independente



Obrigada

Prof. Eliane Faveron Maciel

UNIFACS
ecossistema ânima

25 de março de 2024