Professor: Daniel Tadeu Petinice
E-mail: daniel.petinice@sp.senai.br

Levantamento de Requisitos

Contexto:

Os requisitos de software são fundamentais para orientar o desenvolvimento de projetos, detalhando o que o sistema deve fazer (requisitos funcionais) e como deve ser em termos de desempenho, segurança, e outros aspectos qualitativos (requisitos não funcionais). Documentos como SRS, casos de uso, histórias de usuários, e DFDs são essenciais para a comunicação clara desses requisitos. Eles servem de base para o planejamento, design, implementação e teste, garantindo que o produto final esteja alinhado com as expectativas dos stakeholders (partes interessadas).

1. Requisitos

Os requisitos são descrições detalhadas do que um sistema deve fazer, servindo como uma ponte entre as necessidades dos usuários finais e as funcionalidades que serão implementadas. Eles são essenciais para o desenvolvimento de projetos de software, pois orientam as equipes de desenvolvimento sobre o que precisa ser construído.

No desenvolvimento de software, os requisitos funcionam como uma bússola, orientando a equipe desde a concepção até a implementação do projeto. Eles são declarações detalhadas sobre o que o sistema é esperado fazer e como deve se comportar em diversas circunstâncias.

Ao identificar claramente as necessidades dos usuários e as funcionalidades necessárias para atendê-las, os requisitos ajudam a evitar desvios, garantindo que o produto final esteja alinhado com as expectativas dos stakeholders. Eles são fundamentais para a definição do escopo, para o planejamento do projeto e para a comunicação efetiva entre todos os envolvidos.

2. Definição

A definição de requisitos envolve o processo de **coletar, analisar e definir as necessidades e expectativas dos stakeholders** (partes interessadas) em relação a um novo ou alterado produto. Este processo é crucial para garantir que o produto final atenda às necessidades dos usuários. A definição de requisitos é o alicerce para o sucesso de um projeto de software, agindo como uma ponte entre as expectativas dos stakeholders e as funcionalidades do produto final.

Envolve etapas de coleta, análise e formalização das necessidades dos usuários, garantindo que o desenvolvimento esteja alinhado com os objetivos do projeto. Este processo não apenas identifica o que o sistema deve fazer, mas também ajuda a priorizar funcionalidades, facilitando o planejamento e a execução do projeto de forma eficaz.

Professor: Daniel Tadeu Petinice
E-mail: daniel.petinice@sp.senai.br

Exemplo:

Um exemplo prático é a criação de uma plataforma de e-commerce, onde os requisitos podem variar desde funcionalidades básicas de adicionar produtos ao carrinho até requisitos complexos de integração com sistemas de pagamento.

3. Modelos de documentação

Modelos de documentação de requisitos, como: documentos de especificações de requisitos de software (SRS), casos de uso, histórias de usuários e diagramas de fluxo de dados, são fundamentais para a clareza e a eficiência na comunicação de requisitos dentro de um projeto.

Eles servem como ferramentas padronizadas que ajudam a equipe a entender, de forma unificada, o **que precisa ser desenvolvido**, minimizando ambiguidades e garantindo que todos os aspectos do sistema sejam adequadamente documentados e compreendidos. A escolha do modelo adequado depende das especificidades do projeto e das preferências da equipe.

3.1. Documentos de Especificações de Requisitos de Software (SRS)

Documentos de Especificações de Requisitos de Software (SRS) são documentos formais que detalham todas as funcionalidades, especificações técnicas, requisitos funcionais e não funcionais e outras necessidades que um software deve atender.

Eles servem como um acordo entre a equipe de desenvolvimento e os stakeholders, garantindo que todos tenham uma compreensão clara do que o software deve fazer, além de **servir como uma base para planejamento, design, implementação e testes do projeto**.

Exemplo:

Imagine um SRS para um aplicativo de gerenciamento de eventos. Ele especificaria detalhes como a capacidade de usuários criarem e gerenciarem eventos, convidarem participantes via email. Incluiria requisitos técnicos, como compatibilidade com dispositivos móveis, integração com calendários e medidas de segurança para proteger dados dos usuários. Além disso, definiria critérios de desempenho, como tempos de resposta do sistema e requisitos não funcionais, como usabilidade e acessibilidade. Este documento garantiria que todas as partes tenham expectativas alinhadas sobre o produto final.

Documento de especificação de software:

I. Introdução

1. Referências do Sistema
2. Descrição Geral
3. Restrições de projeto do software

II. Descrição da Informação

1. Representação do fluxo de informação

a. Fluxo de Dados
b. Fluxo de Controle
2. Representação do conteúdo de informação
3. Descrição da interface com o sistema

III Descrição Funcional

1. Divisão funcional em partições
2. Descrição funcional
   a. Narrativas
   b. Restrições/limitações
   c. Exigências de desempenho
   d. Restrições de projeto
   e. Diagramas de apoio
3. Descrição do controle
   a. Especificação do controle
   b. Restrições de projeto

IV. Descrição Comportamental

1. Estados do Sistema
2. Eventos e ações

V. Critérios de Validação

1. Limites de desempenho
2. Classes de testes
3. Reação esperada do software
4. Considerações especiais

VI. Bibliografia
VII Apêndice

3.2. Casos de Uso

Casos de uso são uma técnica usada no desenvolvimento de software para capturar requisitos funcionais, descrevendo as interações entre os usuários (atores) e o sistema para alcançar um objetivo específico.

Eles ajudam a entender como o sistema deve se comportar, apresentando cenários que detalham o fluxo de eventos em várias condições, facilitando a comunicação entre stakeholders técnicos e não técnicos e garantindo que todos os requisitos do usuário sejam atendidos.

Exemplo:

Imagine um caso de uso para um sistema de compras online, onde o objetivo é "***Realizar uma Compra***". Este cenário incluiria passos como o usuário navegando pelo catálogo, selecionando itens para o carrinho, fornecendo informações de pagamento e confirmando a compra. Aqui, o ator principal é o usuário (comprador), e o sistema inclui subcomponentes como o catálogo, carrinho de compras e sistema de pagamento. Esse caso de uso ajuda a equipe de desenvolvimento a entender exatamente como o usuário interage com o sistema durante a compra.

Professor: Daniel Tadeu Petinice
E-mail: daniel.petinice@sp.senai.br

Tabela: Conceitos envolvidos na descrição de um caso de uso.

Objetivo:	Contém uma breve descrição do objetivo do caso de uso.
Requisitos:	Neste campo indicamos a qual requisito funcional o caso de uso em questão está associado.
Atores:	Neste campo definimos a lista de atores associados ao caso de uso. Ator é qualquer entidade externa que interage com o sistema (neste caso, com o caso de uso em questão).
Prioridade:	Informação identificada junto ao usuário que auxilia na definição dos casos de uso que serão contemplados em cada iteração do desenvolvimento do software.
Pré-condições:	Neste campo devemos informar as condições que devem ser atendidas para que o caso de uso possa ser executado.
Frequência de uso:	Informação identificada junto ao usuário que auxilia na definição dos casos de uso que serão contemplados em cada iteração do desenvolvimento do software.
Criticalidade:	Informação identificada junto ao usuário que auxilia na definição dos casos de uso que serão contemplados em cada iteração do desenvolvimento do software.
Condição de Entrada:	Neste campo definimos qual ação do ator dará início à interação com o caso de uso em questão.
Fluxo Principal:	Esta é uma das seções principais do caso de uso. É onde descrevemos os passos entre o ator e o sistema. O fluxo principal é o cenário que mais acontece no caso de uso e/ou o mais importante.
Fluxo Alternativo:	Fluxo alternativo é o caminho alternativo tomado pelo caso de uso a partir do fluxo principal, ou seja, dada uma condição de negócio o caso de uso seguirá por outro cenário que não o principal caso essa condição seja verdadeira.

Professor: Daniel Tadeu Petinice
E-mail: daniel.petinice@sp.senai.br

3.3. História do Usuário

Histórias de usuários são uma técnica ágil para capturar requisitos de software sob a perspectiva do usuário final. Elas descrevem uma funcionalidade que o usuário necessita, geralmente seguindo o formato "Como [tipo de usuário], quero [ação ou funcionalidade] para que [benefício ou valor]".

Essa abordagem foca nos objetivos e necessidades do usuário, facilitando a comunicação entre a equipe de desenvolvimento e stakeholders, e ajuda a priorizar o trabalho em torno do valor entregue ao usuário.

Exemplo:

Considere uma história de usuário para um aplicativo de lembretes: "Como um usuário ocupado, quero ser capaz de definir lembretes por voz, para que eu possa rapidamente criar novos lembretes sem precisar digitar". Essa história ajuda a equipe a compreender a necessidade de uma interface de voz amigável e eficaz, focando no benefício de economizar tempo e aumentar a conveniência para o usuário.

As histórias do usuário em geral são expressas em uma frase simples, como as seguintes: "Como [persona], eu [quero], [para que]."

Detalhando:

"Como [persona]": para quem estamos criando? Não estamos em busca de apenas um cargo, estamos em busca da persona da pessoa.

"Quer": aqui descrevemos a intenção da pessoa, não os recursos que ela usa. O que ela quer alcançar. Esta declaração não deve tratar de implementação – se estiver descrevendo qualquer parte da IU e não a meta do usuário, você entendeu errado.

"Para que": como a vontade imediata dele de fazer algo se encaixa no cenário geral? Qual é o benefício geral que ele quer alcançar? Qual é o grande problema que precisa de solução?

Exemplo, é possível escrever histórias do usuário assim:

- Como Max, eu quero convidar meus amigos, para que a gente possa aproveitar este serviço juntos.
- Como Sascha, eu quero organizar meu trabalho, para que eu me sinta mais no controle.
- Como gerente, eu quero conseguir entender o progresso dos meus colegas, para que eu possa ter relatórios melhores dos nossos acertos e falhas.

Professor: Daniel Tadeu Petinice
E-mail: daniel.petinice@sp.senai.br

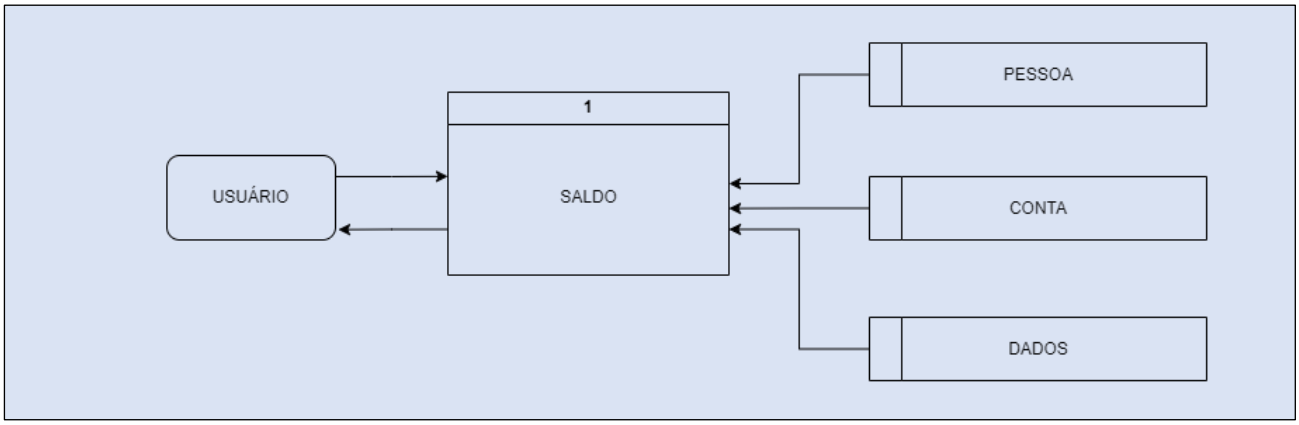
3.4. Diagramas de Fluxo de Dados (DFD)

Diagramas de Fluxo de Dados (DFD) são uma representação gráfica dos fluxos de dados através de um sistema de informação. Eles mapeiam como os dados são processados e armazenados, mostrando a origem, o destino, o armazenamento de dados e os processos que transformam os dados de entrada em saída.

DFDs ajudam a entender a visão geral do sistema, facilitando a comunicação entre analistas e stakeholders e a identificação de ineficiências ou redundâncias nos processos de negócios.

Exemplo:

Imagine um DFD para um aplicativo de banco online, onde um usuário deseja verificar seu saldo. O diagrama mostraria dados fluindo do usuário (solicitando saldo) para o sistema, que então acessa o banco de dados de contas, recupera as informações de saldo e as envia de volta para a interface do usuário. Este DFD ajudaria a equipe a visualizar o processo de consulta de saldo, identificando pontos onde os dados são processados, armazenados e como a informação retorna ao usuário.



4. Regras de Negócio

Regras de negócio são diretrizes que governam as operações, definições e restrições dentro de um ambiente de negócios. Elas são usadas para garantir que o comportamento do sistema esteja alinhado com as metas, estratégias e políticas da organização. As regras de negócio precisam ser claras e bem definidas para que os requisitos possam refletir adequadamente como o sistema deve operar dentro desses parâmetros. As regras de negócio são essenciais para alinhar o desenvolvimento de sistemas às metas e estratégias organizacionais. Elas definem como as operações e transações devem ocorrer dentro de um sistema, influenciando diretamente no desenho e na funcionalidade do software. Um exemplo prático seria um sistema bancário que, seguindo as regras de negócio, determina os tipos de transações permitidas para diferentes tipos de contas, como saques, depósitos e transferências, garantindo a aderência às políticas internas e regulamentações do setor.

Professor: Daniel Tadeu Petinice
E-mail: daniel.petinice@sp.senai.br

Exemplo:

Conversor de Moedas (desenvolvido em sala de auala)

Imagine que você faz parte de uma equipe de estudantes de programação que foi desafiada a criar um projeto prático. O projeto consiste no desenvolvimento de um Conversor de Moedas, especificamente para converter valores em Dólar (USD) para Real (BRL). Esse programa será especialmente útil para estudantes de intercâmbio e viajantes que precisam calcular despesas em diferentes moedas. Sua tarefa é desenvolver um programa que possa converter eficientemente valores em Dólar para Real e vice-versa, considerando a taxa de câmbio atual. O programa deve ser fácil de usar, preciso e fornecer resultados claros.

No exercício do Conversor de Moedas, as regras de negócio podem ser vistas na forma como o programa trata a conversão de moedas.

Primeiro, ele solicita a taxa de câmbio atual, que é uma variável essencial no cálculo de conversão. A escolha entre converter de Dólar para Real ou vice-versa permite que o programa atenda a diferentes necessidades do usuário, alinhando-se às regras de negócio ao fornecer flexibilidade e precisão. Além disso, o arredondamento do resultado final para duas casas decimais reflete uma prática comum em transações financeiras, garantindo que o valor convertido seja prático e facilmente compreensível.

5. Restrições

Restrições são limitações ou condições que o sistema deve satisfazer ou operar dentro delas. Podem incluir limitações tecnológicas, regulamentações legais, limitações de recursos e requisitos de compatibilidade. As restrições são importantes para definir o escopo do projeto e garantir que as soluções propostas sejam viáveis e realistas. Restrições em um projeto de software definem os limites dentro dos quais a solução deve operar, abrangendo desde limitações tecnológicas e regulamentações legais até restrições de recursos e requisitos de compatibilidade. Elas são cruciais para delinear o escopo do projeto, assegurando a viabilidade e realismo das soluções propostas. Por exemplo, um aplicativo desenvolvido para iOS deve aderir às diretrizes da Apple e ser compatível com as versões específicas do sistema operacional.

Exemplo do conversor de moedas:

As restrições do projeto de um Conversor de Moedas podem incluir a necessidade de uma taxa de câmbio atual e confiável, a compatibilidade do programa em diferentes plataformas ou dispositivos (caso fosse além de Portugol), e a precisão no cálculo e apresentação dos resultados, considerando o arredondamento para duas casas decimais. A eficiência na conversão e a facilidade de uso também são restrições importantes, direcionando o design da interface e a interação do usuário com o programa.

Professor: Daniel Tadeu Petinice
E-mail: daniel.petinice@sp.senai.br

6. Tipos de Requisitos

6.1. Funcionais

Requisitos funcionais descrevem o que o sistema deve fazer, ou seja, as funções e características específicas que o software deve possuir. Eles se referem diretamente às interações do usuário com o sistema, como processamento de dados, operações lógicas, e como o sistema deve responder a determinadas entradas. Os requisitos funcionais são o coração de qualquer sistema, descrevendo as ações específicas e operações que o software deve ser capaz de executar. Eles detalham cada funcionalidade, como cadastro de usuários, processamento de pagamentos e geração de relatórios, fornecendo um guia claro para o desenvolvimento. Por exemplo, em um aplicativo bancário, um requisito funcional poderia ser "o sistema deve permitir que o usuário transfira dinheiro entre contas internas em menos de 3 minutos".

Exemplo:
No projeto Conversor de Moedas, os requisitos funcionais incluem a habilidade de o sistema solicitar a cotação atual do dólar, permitir ao usuário escolher entre converter dólar para real ou real para dólar, receber o valor a ser convertido e calcular o resultado baseado na cotação fornecida. Essas funcionalidades específicas garantem que o programa atenda às necessidades dos usuários de forma eficiente, fácil de usar e com resultados precisos.

6.2. Não funcionais

Requisitos não funcionais, por outro lado, descrevem como o sistema deve ser, ou seja, atributos relacionados ao desempenho, segurança, interface do usuário, compatibilidade, e outras propriedades de qualidade. Eles não estão diretamente ligados às funções específicas do sistema, mas afetam a experiência do usuário e a eficiência do sistema como um todo. Requisitos não funcionais enfatizam a qualidade e os critérios operacionais que o sistema deve cumprir, como desempenho, segurança, usabilidade e compatibilidade. Eles são fundamentais para garantir a satisfação do usuário e a eficiência do sistema, mas não descrevem funcionalidades diretas. Por exemplo, para um website de e-commerce, um requisito não funcional poderia ser "a página de produto deve carregar em menos de 2 segundos em conexões de banda larga".

Exemplo:
Por exemplo, no contexto do Conversor de Moedas, poderiam incluir a rapidez na conversão, a segurança na manipulação de dados do usuário e a adaptabilidade do programa a diferentes dispositivos e sistemas operacionais.

Professor: Daniel Tadeu Petinice
E-mail: daniel.petinice@sp.senai.br

Referências:

1. Monitoratec. "Especificação de Requisitos de Software". Disponível em: https://www.monitoratec.com.br/blog/especificacao-de-requisitos-de-software/. Acesso em: 01/03/2024.

2. Visure Solutions. "Requirements Specification". Disponível em: https://visuresolutions.com/pt/blog/requirements-specification/. Acesso em: 01/03/2024.

3. Universidade Federal de Santa Catarina (UFSC). "Norma ISO 29110 Perfil de Entrada Básico". Disponível em: https://www.inf.ufsc.br/~jean.hauck/guias/29110/Norma%20ISO%2029110%20Perfil%20de%20Entrada%20B%C3%A1sico/guidances/examples/documento_especificao_de_requisitos_2CFB343D.html. Acesso em: 01/03/2024.

4. DevMedia. "Especificação de Requisitos com Casos de Uso". Disponível em: https://www.devmedia.com.br/especificacao-de-requisitos-com-casos-de-uso/10245. Acesso em: 01/03/2024.

5. Atlassian. "User Stories". Disponível em: https://www.atlassian.com/br/agile/project-management/user-stories. Acesso em: 01/03/2024.

6. Lucidchart. "O que é um Diagrama de Fluxo de Dados". Disponível em: https://www.lucidchart.com/pages/pt/o-que-e-um-diagrama-de-fluxo-de-dados. Acesso em: 01/03/2024.

7. SOMERVILLE, Ian. Engenharia de software. São Paulo: Pearson Prentice Hall, 2011.

8. FOWLER, M. UML Essencial: Um breve guia para a linguagem-padrão de modelagem de objetos. Porto Alegre: Bookman, 2005.

9. GANE, C.; Sarson, T. Análise estruturada de sistemas. Rio de Janeiro: Livros Técnicos e Científicos. Editora S.A., 1983.