AULA 1 – CONCEITOS DE REQUISITOS

OBJETIVO DA AULA

Esta aula tem o objetivo de apresentar a conceituação sobre requisitos de software e relacionar a importância da elicitação, validação e documentação dos requisitos no processo de construção do software.

APRESENTAÇÃO

Os requisitos são necessidades que um software precisa cumprir, para que atenda os objetivos do cliente.

Os requisitos possuem diversos tipos, principalmente agrupados em funcionais e não funcionais, dentro dessa divisão eles têm outros subtipos que ajudam a identificar as conformidades mais importantes de um software.

Nesta aula aprenderemos o que é um requisito e entenderemos como funciona o seu ciclo de vida, focando em cada etapa, e nas atividades realizadas em cada uma dessas fases.

1. CONCEITOS E REQUISITOS

O processo de desenvolvimento de software é composto de várias fases, que podem variar conforme o processo de software escolhido. Mas se há uma questão que comum a todos eles, esta é a Engenharia de Requisitos.

A Engenharia de requisitos tem o objetivo de definir um conjunto de atividades e técnicas que passam pelas fases de Elicitação ou Levantamento de Requisitos, Análise e Negociação de Requisitos, Especificação e Documentação de Requisitos, Verificação e Validação de Requisitos e Gerenciamento de Requisitos. Mas o que são Requisitos?

Requisitos são características que o usuário necessita para resolver um problema ou atingir um objetivo; ou ainda uma característica que o sistema deve possuir ou atingir para satisfazer um contrato (*Institute of Electrical and Electronics Engineers*, 1990, p. 84).

Ou seja, quando um cliente define as funcionalidades de um sistema e as suas restrições, ele está definindo os requisitos necessários para a solução encomendada.

O processo da Engenharia de Requisitos prevê um ciclo de vida que obedece a um fluxo de trabalho em que várias questões são observadas, entendidas e documentadas para que esse processo seja o mais completo e transparente possível.

Na prática, a Engenharia de Requisitos abordará essencialmente:

- Elicitação consistente de requisitos do Software;
- Qualidade do Software que está sendo produzido;
- Produtividade na gestão do desenvolvimento do Software;
- Gerenciamento de prazos, orçamento e necessidades do cliente;
- Produção de documentação do sistema.

2. CICLO DE VIDA DA ENGENHARIA DE REQUISITOS

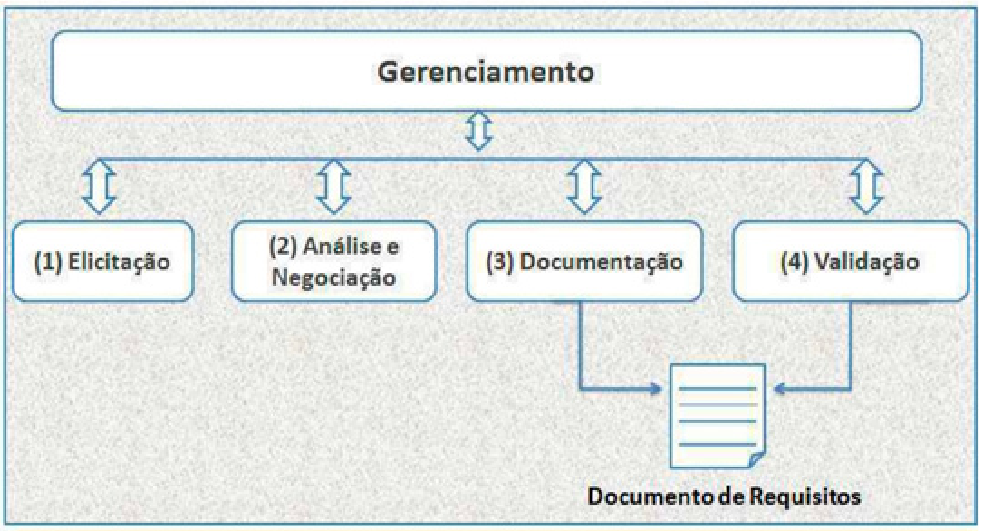
Estima-se que 40% do percentual de erros identificados nos sistemas é causado por falha no processo de especificação de requisitos malfeita. E isso traz um grande problema em termos de custo de manutenção, já que é muito mais caro reparar um erro de especificação de requisitos do que um erro em fases como a codificação ou projeto de interface.

O objetivo da Engenharia de Requisitos é apoiar a construção de software com qualidade, e o conceito de qualidade é bem simples: satisfazer as necessidades dos clientes dentro do prazo e do orçamento estabelecido.

O ciclo de vida da Engenharia de Requisitos é composto por 5 fases:

- Elicitação ou Levantamento de Requisitos;
- Análise e Negociação de Requisitos;
- Especificação ou Documentação de Requisitos;
- Verificação e Validação de Requisitos;
- Gerenciamento de Requisitos.

FIGURA 1 | **Ciclo de Vida da Engenharia de Requisitos**



Fonte: Adaptado de (THAYER; DORFMAN, 1997). Disponível em: https://ri.ufs.br/bitstream/riufs/6886/2/Fiama%20da%20Silva%20Santos.pdf. Acesso em: 19 nov. 2022.

A Figura 1 mostra um exemplo de ciclo de vida de Engenharia de Requisitos, em que cada fase é sequenciada com o gerenciamento de requisitos acompanhando cada ação realizada.

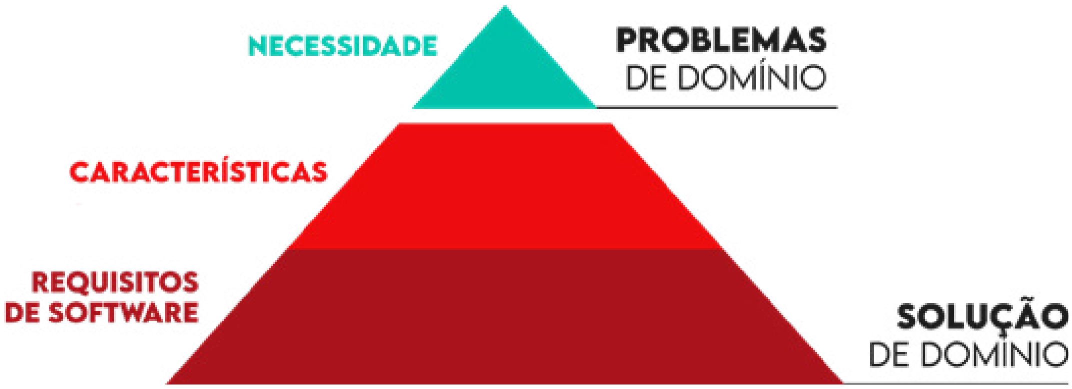
Agora vamos discutir quais são as atividades realizadas em cada fase da Engenharia de Requisitos:

2.1. ELICITAÇÃO OU LEVANTAMENTO DE REQUISITOS

Elicitar Requisitos significa levantar e identificação problemas, para a busca de uma solução tecnológica, partindo da análise das necessidades dos usuários e do negócio.

Nesta fase o objetivo é descobrir quais são as necessidades do cliente, ou seja, as aspirações e expectativas que o cliente gostaria que fossem satisfeitas pelo novo sistema.

FIGURA 2 | **Pirâmide de Requisitos**



Fonte: Adaptado de Leffingwell and Widrig 2003, p. 21. Disponível em https://www.researchgate.net/figure/Requirements-Pyramid-adapted-from-Leffingwell-and-Widrig-2003-p-21_fig15_306012386. Acesso em: 19 nov. 2022.

A Figura 2 mostra a Pirâmide de Requisitos, que descreve como as necessidades se transformam em Requisitos de Software.

Podemos citar como exemplo de necessidade, a facilitação do processo de pedir uma pizza por parte do cliente, que no cenário atual de uma suposta pizzaria, precisa telefonar para realizar o seu pedido.

Uma necessidade pode ser atendida a partir de uma nova funcionalidade de um sistema, por exemplo, "Realizar Pedido de Pizza Via Aplicativo".

Essa funcionalidade cumprirá requisitos que possuem aspectos que devem ser atendidos, como agilidade, facilidade no uso, segurança, e outros relativos especificamente à funcionalidade que está sendo executada, como, por exemplo, escolher sabor da pizza, aplicar promoção, realizar pagamento, incrementar programa de fidelidade etc.

Simplificando, a Elicitação de Requisitos visa entender qual é o problema a ser resolvido, utilizando uma série de técnicas de elicitação no apoio deste processo.

2.2. ANÁLISE E NEGOCIAÇÃO DE REQUISITOS

Nesta fase os requisitos descobertos na fase anterior devem ser analisados e, com os *stakeholders*, devem-se identificar os conflitos, omissões, inconsistências e ambiguidades, no sentido de garantir a produção de uma lista documentada onde já se observou a viabilidade de atendimento dos requisitos, já foram propostas de adaptações e em que foram negociadas alternativas de solução.

2.3. ESPECIFICAÇÃO E DOCUMENTAÇÃO DE REQUISITOS

Como o próprio nome sugere, essa fase tem o objetivo de documentar os requisitos para que a comunicação entre a equipe de desenvolvimento seja favorecida, de forma que todos tenham um entendimento claro sobre o sistema.

A documentação também vai garantir os processos de rastreabilidade e auditabilidade dos requisitos, deve-se então ser elaborado um documento de requisitos visando deixar claro o que será implementado no sistema.

Documentar os requisitos em um artefato próprio facilita o controle de alterações de todos os envolvidos na manutenção dos requisitos, permitindo também a como a geração de versões do documento e a disponibilização do documento para os envolvidos no desenvolvimento do produto.

2.4. VERIFICAÇÃO E VALIDAÇÃO DOS REQUISITOS

Nesta fase objetiva-se garantir a consistência das descrições dos requisitos, isto é, como cada requisito deverá ser executado, para não haver nenhum problema de entendimento sobre os requisitos.

Os *stakeholders* validam este documento com o objetivo de eliminar qualquer tipo de erro de entendimento.

A validação é a atividade em que obtemos o aceite do cliente sob a lista de requisitos que foram especificados.

2.5. GERENCIAMENTO DE REQUISITOS

A fase de Gerenciamento de Requisitos ocorre paralelamente a todas as outras fases. O objetivo é gerenciar as mudanças nos requisitos que podem ocorrer ao longo de todo o processo, para que qualquer mudança seja feita de maneira controlada.

O gerenciamento de requisitos garante que uma possível mudança seja documentada, para que possam ser identificadas as características desta mudança como quem a solicitou, quem a aprovou, datas, e como esta mudança impacta outras partes no sistema.

Esta fase também visa definir quais métodos serão utilizados ao longo do processo de requisitos, além de fornecer ferramentas que estabeleçam o apoio às atividades da Engenharia de Requisitos.

Em resumo, o processo de mudança em um software deve observar:

- A necessidade de se realizar, de fato, a mudança;
- Identificar quais outros requisitos serão afetados com a mudança;
- A validação desta mudança com os *stakeholders*;
- Os custos que a mudança gerará;
- O impacto da mudança no sistema como um todo.

CONSIDERAÇÕES FINAIS

Nesta aula abordamos um fator essencial na construção de software com qualidade: a engenharia de requisitos.

Gerenciar o processo de requisitos garante que cada fase deste ciclo de vida seja observada com cuidado, e que os melhores resultados sejam obtidos a partir desta prática.

Discutimos o quão crítico é identificar problemas relativos à má especificação de requisitos após o desenvolvimento do software. Para evitar falhas como essa, há o ciclo de vida da Engenharia de Requisitos, composto pelas fases de Elicitação, Análise, Especificação, Validação e Gerência de Requisitos, onde cada uma delas traz estratégias essenciais para a condução do processo.

MATERIAIS COMPLEMENTARES

Vídeo: *O que é Engenharia de Requisitos?* Disponível em: https://www.youtube.com/watch?v=i5WR9xFKg70. Acesso em: 19 nov. 2022.

Vídeo: *Esqueça isso e o seu projeto estará condenado!* Disponível em: https://www.youtube.com/watch?v=rVbJ7ykuLig. Acesso em: 19 nov. 2022.

Link: *Introdução à Engenharia de Requisitos*. Disponível em: https://www.devmedia.com.br/introducao-a-engenharia-de-requisitos/8034. Acesso em: 19 nov. 2022.

REFERÊNCIAS

PRESSMAN, R.G. *Engenharia de Software.* 9ª ed. Rio de Janeiro: McGraw-Hill, 2021.

SOMMERVILLE, I. *Engenharia de Software.* 10ª ed. São Paulo: Pearson Addison. Wesley, 2019.

AULA 2 – TIPOS DE REQUISITOS

OBJETIVO DA AULA

Conhecer os tipos de requisitos e como eles são classificados em um projeto de software.

APRESENTAÇÃO

Os requisitos são as características que um software deve apresentar para atender as necessidades de um cliente. Eles podem ser classificados a partir de suas características.

A classificação de requisitos funcionais e não funcionais são as mais comuns, mas há outras formas de definir um requisito e o que ele deve garantir em um projeto de software.

Nesta aula aprenderemos a realizar essa classificação, abordando quais são as necessidades que tais requisitos devem cumprir.

1. TIPOS DE REQUISITOS

O entendimento dos requisitos é essencial para a construção de um software com qualidade. Basicamente os requisitos são divididos em Funcionais e Não-Funcionais.

Os Requisitos Funcionais são aqueles que dizem respeito às funcionalidades do sistema, ou seja, aquilo que é executável. Uma declaração de requisitos pode trazer também a identificação do que o sistema NÃO deve fazer, para que isso fique explícito para os *stakeholders*.

Em um sistema de uma pizzaria, por exemplo, poderíamos ter como requisitos funcionais os seguintes requisitos:

- Cadastrar Cliente;
- Cadastrar Pedido;
- Aplicar Cupom de Desconto;
- Cadastrar Pizza.

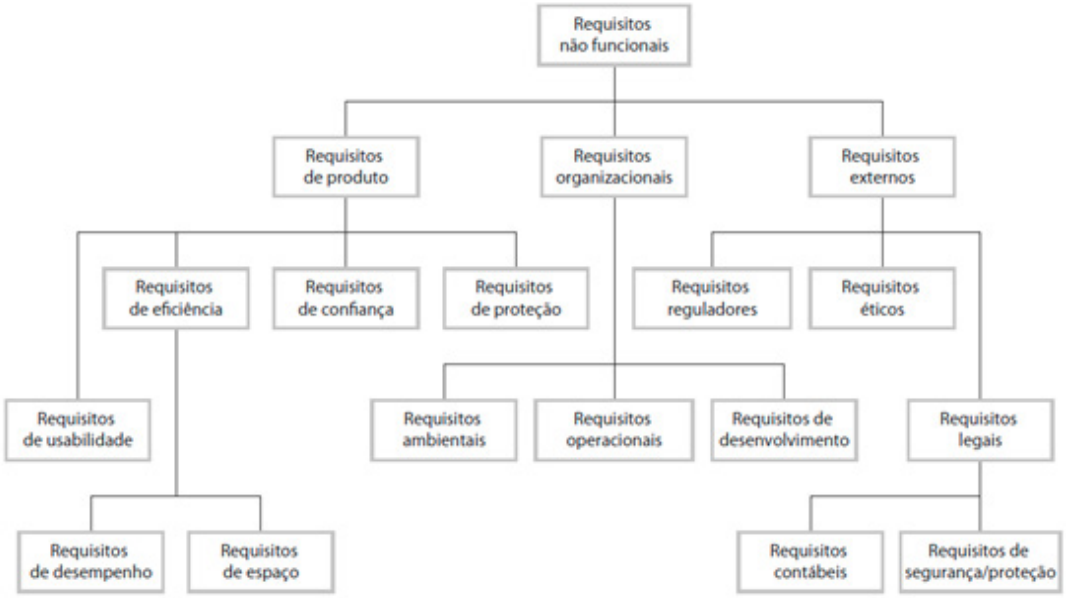
Já os Requisitos Não-Funcionais dizem respeito às condições que um sistema deve atender, ou seja, rapidez, segurança, usabilidade, confidencialidade, confiabilidade, regulação, dentre outros que não especifiquem uma funcionalidade. Eles definem as restrições de funcionamento do sistema.

Considerando o hipotético sistema de uma pizzaria, alguns requisitos funcionais do sistema poderiam ser:

- O sistema deverá exibir o cardápio em, no máximo, 2 segundos;
- O aplicativo da pizzaria deverá ser responsivo;
- Os dados do cartão de crédito não devem estar acessíveis.

De acordo com Sommerville (2011, p. 61), os requisitos não funcionais podem ser provenientes das características requeridas para o software (requisitos de produto), da organização que desenvolve o software (requisitos organizacionais) ou de fontes externas:

FIGURA 1 | **Diagrama de Requisitos Não-Funcionais**



Fonte: Sommerville (2011, p. 61).

Na Figura 1 vemos as possíveis ramificações que os Requisitos Funcionais podem ter, o que facilita o entendimento de que muitas características podem ter influência no desenvolvimento de um sistema, para que as especificações sejam atendidas.

Tendo em vista essas ramificações, podemos afirmar que os requisitos são caracterizados da seguinte forma:

- Requisitos de Produto: tem a ver com o funcionamento do software, diz respeito ao desempenho, rapidez, uso de memória, confiabilidade, usabilidade e segurança, por exemplo;
- Requisitos Organizacionais: vão listar características inerentes às políticas e procedimentos da organização e do desenvolvedor, incluindo questões como sistema operacional, linguagem de programação e normas gerais da organização;
- Requisitos Externos: apresentam fatores externos que o sistema deve cumprir, como normas reguladoras, requisitos legais e requisitos éticos.

Quanto aos requisitos não funcionais, é importante que eles sejam especificados quantitativamente, para que ele seja testável e que não gere interpretações equivocadas ou dúbias.

Por exemplo, podemos especificar um requisito funcional da seguinte forma:

O sistema deverá realizar o backup em, no máximo, cinco minutos.

Para ajudar no processo de especificação de requisitos não funcionais, Sommerville (2011, p. 63) sugere as seguintes métricas dispostas na Tabela 1:

TABELA 1 **| Métricas para Requisitos Não-Funcionais**

Requisito	Medida
Velocidade	Transações processadas/segundo Tempo de resposta de usuário/evento Tempo de atualização de tela
Tamanho	Megabytes/Gigabytes
Usabilidade	Tempo de Treinamento
Confiabilidade	Tempo médio para falha Probabilidade de indisponibilidade Taxa de ocorrência de falhas Disponibilidade
Robustez	Tempo de reinício após falha Percentual de eventos que causam falhas Probabilidade de corrupção de dados em caso de falha
Portabilidade	Percentual de declarações dependentes do sistema-alvo Número de sistemas-alvo

Fonte: Adaptada de Sommerville (2011, p. 63).

Os Requisitos Não-Funcionais podem ainda se basear na sigla UCDS (Confiabilidade, Desempenho e Suportabilidade), que determinam as características desejáveis do software.

Em termos práticos, cada característica dessas tem o objetivo de:

- Usabilidade: tratar da facilidade com que o software pode ser utilizado pelos usuários;
- Confiabilidade: tratar da habilidade do sistema de se comportar de forma consistente e aceitável;
- Desempenho: tratar da velocidade ou eficiência de execução do sistema;
- Suportabilidade: tratar da habilidade do software em ser facilmente modificável para acomodar melhorias e reparos ao longo de sua operação.

2. OUTROS TIPOS DE REQUISITOS

Em um contexto mais amplo, podemos dizer que todos os requisitos são requisitos de projeto, e ainda que eles possam ter subdivisões que podem ser assim representadas:

a) Requisitos de Projeto:

- Requisitos de Marketing;

- Requisitos de Hardware;
- Requisitos de Produção;
- Requisitos de Processos Organizacionais;
- Requisitos de Software:
- Requisitos Funcionais;
- Requisitos Não-Funcionais.

Exemplos:

- Requisitos de Marketing – Aponta um direcionamento do produto, por exemplo: o sistema deverá ser direcionado para estudantes de doutorado;
- Requisitos de Hardware – Uma condição de hardware para uso do sistema, por exemplo: o sistema precisa de um processador, no mínimo, i7 ou equivalente;
- Requisitos de Produção – Direciona o processo de produção, por exemplo: o sistema deverá ser desenvolvido utilizando o método ágil Scrum;
- Requisitos de Processos Organizacionais – Aborda processos e regras de negócio da organização, por exemplo: o sistema deverá permitir o cadastro de atividades de alunos monitores.

3. DOCUMENTO DE REQUISITOS

Após a identificação de requisitos, o próximo passo é documentá-los. A documentação de requisitos é especialmente importante porque ela define o escopo do sistema, isto é, tudo aquilo que o sistema deverá apresentar ao seu término.

A literatura da **Engenharia de Software** propõe a construção de documentos guias para que o analista possa documentar e especificar os requisitos descobertos. No entanto, não há a necessidade de seguir o padrão apresentado por essa literatura, o mais importante é a padronização deste tipo de documentação em uma organização ou projeto e, além disso, também é importante que o documento seja útil, e ajude aos *stakeholders* a conhecer as funcionalidades que serão implantadas no sistema.

Outro ponto importante é que seja considerado o tempo de produção deste documento, principalmente em casos em que os requisitos mudam rapidamente. Não se deve perder tempo demasiado na construção deste documento, para que isso não impacte no tempo total de elaboração do projeto.

Como boa prática, um documento de requisitos deve conter:

- Índice – para ajudar a organizar o trabalho e para que o leitor encontre as informações desejadas;
- Histórico de Versões – para controlar as mudanças no documento;

- Introdução – para descrever o sistema de forma geral;
- Glossário – para descrever os termos técnicos utilizados no projeto;
- Diagrama de Caso de Uso – trata-se de um diagrama da UML (Linguagem de Modelagem Unificada) que deve exibir os requisitos funcionais do sistema;
- Requisitos Funcionais e Não-Funcionais – deve-se listar e descrever os requisitos funcionais e não funcionais encontrados;
- Especificação ou Descrição dos Requisitos Funcionais e Não-Funcionais – para detalhar como os requisitos funcionais e não funcionais devem funcionar dentro do sistema.

Novamente, é importante dizer que isso é apenas uma sugestão de como elaborar um documento de requisitos, mas este artefato pode trazer muitas outras informações que sejam úteis para equipe. A elaboração de um *template* para ser usado pela equipe é outra boa prática. Vale a pena lembrar que, embora esse documento seja muito útil, precisamos tomar cuidado com o tempo que levamos para produzi-lo.

CONSIDERAÇÕES FINAIS

Nesta aula abordamos os tipos de requisitos, que podem ser geralmente classificados em Funcionais e Não-Funcionais, mas vimos também que existem outras formas de classificar os requisitos, segmentando ainda mais o que é desejado pelos *stakeholders*.

Quanto aos requisitos Não-Funcionais, há uma infinidade de classificações, trazendo as características como eficiência, usabilidade e portabilidade, além dos requisitos que tratam de questões legais e regulatórias.

Vimos ainda que a elaboração de um *template* para documentar os requisitos é uma boa prática, já que este documento apoia a organização, a auditabilidade e o rastreio de mudanças ao longo do ciclo de vida do produto.

MATERIAIS COMPLEMENTARES

Link: *Como escrever requisitos de software de forma simples e garantir o mínimo de erros no sistema/app?* Disponível em: https://medium.com/lfdev-blog/como-escrever-requisitos-de-software-de-forma-simples-e-garantir-o-m%C3%ADnimo-de-erros-no-sistema-app-74df2ee241cc. Acesso em: 23 nov. 2022.

Vídeo: *Entenda a diferença entre Requisitos Funcionais e Não-Funcionais*. Disponível em: https://www.youtube.com/watch?v=YLd6AWKVyas. Acesso em: 23 nov. 2022.

Vídeo: *Qual é a melhor forma de documentar software?* Disponível em: https://www.youtube.com/watch?v=S3NLU898_Cc. Acesso em: 23 nov. 2022.

REFERÊNCIAS

PRESSMAN, R.G. *Engenharia de Software*. 9ª ed. Rio de Janeiro: McGraw-Hill, 2021.

SOMMERVILLE, I. *Engenharia de Software*. 10ª ed. São Paulo: Pearson Addison. Wesley, 2019.

AULA 3 – TÉCNICAS DE ELICITAÇÃO E ANÁLISE DE REQUISITOS

OBJETIVO DA AULA

Conhecer as técnicas de elicitação e análise de requisitos.

APRESENTAÇÃO

A Elicitação de Requisitos é um processo crucial na construção de software. Já vimos que um erro de requisito descoberto ao final do projeto, pode comprometer todo o sistema, apresentando altos custos de correção.

Nesta aula vamos abordar alguns dos principais desafios encontrados pelos usuários e desenvolvedores no processo de elicitação de requisitos.

Vamos apresentar também as principais técnicas de elicitação de requisitos disponíveis através da literatura técnica, elencando as suas vantagens, desvantagens e situações de aplicação.

1. ELICITAÇÃO DE REQUISITOS

Elicitação de Requisitos é o processo de descobrir, explicitar e obter o máximo de informações para o conhecimento do produto em questão. Esse processo visa descobrir todas as características do sistema para não haver dúvidas ou inconsistências entre os envolvidos sobre o que o sistema deverá ser capaz de fazer.

Logicamente esse processo não é nada fácil, pois o cliente nem sempre tem a exata noção e conhecimento de seus processos, nem mesmo de como a tecnologia é capaz de apoiá-los.

Para facilitar esse processo, existem as chamadas técnicas de elicitação de requisitos, que não meios executáveis para facilitar o entendimento geral do processo, a partir de aplicação estratégias que visam tornar o processo mais assertivo.

Essas técnicas vão identificar os fatos que compõem os requisitos do sistema, extraindo deles as características fundamentais do futuro produto, provendo entendimento do que software precisa efetivamente fazer.

2. DESAFIOS NA ELICITAÇÃO DE REQUISITOS

Alguns desafios tornam essa fase ainda mais complexa. É notório que os usuários do processo atual possam esquecer pontos importantes, bem como, ao final, identificar que alguns requisitos não foram bem compreendidos pelos analistas, ou ainda que faltam funcionalidades importantes.

Outro problema é a chamada “Síndrome das Ruínas Não Descobertas”, que “quanto mais se descobre, sente-se que há mais por descobrir e, mesmo com os melhores procedimentos de elicitação e especificação, uma série de problemas ainda persiste” (LEFFINGWELL; WIDRIG, 2003). Esse é exatamente um caso angustiante para o analista que acaba pensando no que mais ainda há para ser descoberto.

Há outro embate interessante que envolve os analistas e os usuários, pois a comunicação entre eles nem sempre é fácil. Entendemos essa dificuldade ao relacionar o tipo de atividade, experiência e a forma de se comunicar particular que cada grupo possui. Uma mesma palavra pode ter sentidos diferentes para o cliente e para o analista, e isso pode gerar alguns sérios problemas de entendimento.

3. TÉCNICAS DE ELICITAÇÃO DE REQUISITOS

Como já citamos nesta aula, as técnicas de elicitação visam apoiar os analistas e fazê-los “entrar no mundo” dos usuários e clientes. A literatura apresenta uma série de técnicas que podem ajudar nesse processo.

Uma técnica de elicitação é normalmente usada em conjunto com outra técnica, sendo que muitas técnicas podem ser combinadas para municiar o desenvolvedor com todas as informações possíveis sobre o processo que está sendo analisado.

As principais técnicas existentes na literatura são:

- *Brainstorming*;
- Cenários;
- Entrevistas;
- Garimpo de Documentos;
- Observação ou Etnografia;
- Participação Ativa dos Clientes;
- Prototipagem;
- Questionários;
- *Role Playing*;
- Workshops.

Mas isso não quer dizer que só existem essas formas de se elicitar requisitos, cada profissional, equipe ou organização pode elaborar a sua própria prática para identificar requisitos no ambiente do usuário.

Vamos agora entender um pouco sobre cada técnica, conhecendo as suas vantagens e desvantagens:

3.1. BRAINSTORMING

Técnica utilizada para gerar ideias sobre possíveis soluções sistêmicas. Nesta técnica, cada profissional envolvido apresenta as suas ideias (*brainstorming* significa tempestade de ideias), as melhores e mais factíveis podem ser aprovadas para a elaboração no sistema.

Esta técnica é normalmente aplicada em etapas preliminares do processo como um todo.

a) Vantagens:

- Estimula a criatividade;
- Possibilita a inovação;

b) Desvantagens:

- Pode se perder muito tempo, caso não haja um bom gerenciamento;
- A participação de pessoas que conhecem pouco a rotina de trabalho pode atrapalhar o desenvolvimento da técnica.

3.2. CENÁRIOS

Os cenários são histórias que simulam a execução de vários processos no sistema.

Nesse tipo de técnica, são apurados desdobramento para várias situações, para apresentar soluções sistêmicas para cada uma delas.

a) Vantagens:

- As situações normalmente são de domínio do usuário, o que ajuda a ele se lembrar de como ele resolve cada situação.

b) Desvantagens:

- Necessita de um grande tempo de execução para que todos os cenários possíveis sejam investigados.

3.3. ENTREVISTAS

Esta é a técnica mais usada de todas. Praticamente qualquer projeto vai necessitar de mais de uma entrevista com pessoas diversas envolvidas no projeto.

É uma técnica facilmente aplicável, mas deve-se tomar alguns cuidados, como o de preparar um roteiro para que nenhuma pergunta importante seja esquecida.

Hoje há diversas possibilidades e facilitadores para executar uma entrevista, como a possibilidade de gravação e a realização por meio remoto, o que ajuda bastante na otimização do tempo dos envolvidos.

Esta técnica não é recomendada para ser aplicada em um grande número de pessoas, pois tomaria muito tempo e o resultado pode não ser efetivo.

a) Vantagens:

- Entendimento imediato;
- Contato direto com pessoas-chave no processo.

b) Desvantagens:

- Se mal administrada por ser uma perda de tempo;
- Algumas pessoas podem não se sentir à vontade para dar informações importantes sobre o projeto.

3.4. GARIMPO DE DOCUMENTOS

Nesta técnica, os documentos de processos e regras de negócio relacionadas ao sistema são acessados e lidos. A ideia é recuperar normas, padrões e fluxo de atividades para melhor entender o funcionamento dos processos de negócio.

Esta técnica também contribui para que a equipe de analistas passe a conhecer o vocabulário e os jargões utilizados no dia a dia da empresa.

a) Vantagens:

- Acesso aos documentos oficiais.

b) Desvantagens:

- Grande volume de documentos para a leitura;
- Possibilidade de obter documentos desatualizados.

3.5. OBSERVAÇÃO OU ETNOGRAFIA

Esta técnica é bastante simples, pois consiste na observação do processo acontecendo no local. O analista deve observar e anotar as características observadas no processo, em caso de dúvidas, ele deve questionar o usuário para obter um melhor entendimento.

a) Vantagens:

- Ajuda a resolver problemas de falta de tempo por parte dos usuários;
- Ajuda aos usuários que não conseguem explicar direito o funcionamento de um processo.

b) Desvantagens:

- O usuário pode se sentir constrangido ao ser observado;
- Reduz o campo de observação, caso a técnica seja aplicada a apenas um usuário.

3.6. PROTOTIPAGEM

Técnica que prevê a criação de protótipos para a validação de requisitos e interface junto ao cliente.

Esta técnica é importante quando não se tem ideia de como uma funcionalidade será apresentada e executada em um sistema.

a) Vantagens:

- Validação imediata junto ao cliente;
- Permite ao cliente participar ativamente da elaboração do sistema.

b) Desvantagens:

- Custo de tempo na elaboração dos protótipos.
- Usuário pode acreditar que o protótipo já é uma versão final do produto.
- Usuário se concentrar mais na aparência do que na funcionalidade em si.

3.7. QUESTIONÁRIOS

Esta técnica consiste na aplicação de questionários para serem respondidos por usuários e pessoas importantes no processo. Ela é especialmente útil quando há um grande número de pessoas que devem ser entrevistadas.

Esses questionários podem ser do tipo fechado, quando as respostas são objetivas, bastando ao respondente assinalar as opções; aberto, quando o usuário pode descrever livremente a sua resposta; e misto, quando há perguntas abertas e fechadas.

Hoje há recursos tecnológicos que ajudam na operação do processo e na apuração do resultado, como Google Forms, por exemplo.

a) Vantagens:

- Permite análise estatística, no caso dos questionários do tipo fechados;
- Pode ser aplicada para vários usuários de uma só vez.

b) Desvantagens:

- Interação “fria” com o usuário;

- Se for do tipo aberto, necessita de muito tempo para apurar todas as respostas;
- Limitação na quantidade de perguntas.

3.8. ROLE PLAYING

É uma técnica parecida com a da observação, a diferença é que na *Role Playing*, o próprio analista executa as tarefas, orientado pelo usuário. A ideia nesta técnica é fazer o analista "sentir na pele" o processo acontecendo.

a) Vantagens:

- Ajuda a resolver problemas de falta de tempo por parte dos usuários;
- Ajuda aos usuários que não conseguem explicar direito o funcionamento de um processo.

b) Desvantagens:

- Pode atrapalhar a rotina normal da empresa, por ter alguém sem experiência realizando o processo;
- Reduz o campo de observação, caso a técnica seja aplicada a apenas um usuário.

3.9. WORKSHOPS

Como o próprio nome diz, são reuniões em que o assunto principal é o domínio ou área do sistema que será desenvolvido.

Nessa técnica podem ser trazidos profissionais experientes na área, que já atuaram em processos de desenvolvimento semelhantes, ou que operam sistemas do mesmo tipo, para a troca de informações e aculturamento. Participam também *stakeholders* do cliente e da equipe de desenvolvimento.

Normalmente essa técnica acontece no início do projeto, para que todo o esforço do encontro se reverta em ideias e conhecimento sobre o sistema e o projeto como um todo.

a) Vantagens:

- Acesso a pessoas experiente no assunto;
- Troca de informação entre pessoas de universos distintos.

b) Desvantagens:

- Normalmente possui um custo considerável;
- Precisa de alguém experiente na organização.

Da próxima vez que você participar de um processo de desenvolvimento de software, não faltará ideias para a etapa de elicitação, não é mesmo?

CONSIDERAÇÕES FINAIS

Nesta aula falamos a etapa de elicitação de requisitos, apresentando os desafios e dificuldades enfrentadas pelos analistas para entender quais são as necessidades do cliente.

Abordamos em especial as técnicas de elicitação de softwares existentes, que apoiam usuários e desenvolvedores na correto levantamento de requisitos de sistema.

Identificamos para cada uma das técnicas as suas vantagens e desvantagens, apresentando também os principais cenários de aplicação das mesmas.

Concluímos que, apesar de existirem várias técnicas citadas pela literatura, nada impede que um profissional ou organização pode implementar a sua própria forma de identificar requisitos.

MATERIAIS COMPLEMENTARES

Link: *Ténicas de evantamento de Requisitos*. Disponível em: https://www.devmedia.com.br/tecnicas-para-levantamento-de-requisitos/9151. Acesso em: 2 nov. 2022.

Link: *Reuniões de Levantamento. Como torná-las produtivas?* Disponível em: http://www.linhadecodigo.com.br/artigo/159/reunioes-de-levantamento-como-torna-las-produtivas.aspx. Acesso em: 2 nov. 2022.

REFERÊNCIAS

PRESSMAN, R.G. *Engenharia de Software*. 9ª ed. Rio de Janeiro: McGraw-Hill, 2021.

SOMMERVILLE, I. *Engenharia de Software*. 10ª ed. São Paulo: Pearson Addison. Wesley, 2019.

AULA 4 – MODELAGEM DE REQUISITOS

OBJETIVO DA AULA

Conhecer as formas de modelagem de requisitos utilizando o Diagrama de Caso de Uso, que pertence ao conjunto de diagramas da Linguagem de Modelagem Unificada (UML).

APRESENTAÇÃO

A UML (Linguagem de Modelagem Unificada) é uma linguagem utilizada para modelar aspectos de documentação de sistemas. Ela é composta por 14 diagramas, dentre eles o Diagrama de Casos de Uso.

O Diagrama de Casos de Uso é uma excelente ferramenta visual para demonstrar os requisitos funcionais identificados da fase de elicitação de requisitos.

Nesta aula vamos aprender como utilizar este diagrama para modelar e tornar explícitos os requisitos de um sistema.

1. MODELANDO DE REQUISITOS

Criada por Ivar Jacobson na década de 1980, a técnica chamada *Objectory*, prevê a utilização de Diagramas de Caso de Uso para a modelagem e especificação de requisitos funcionais.

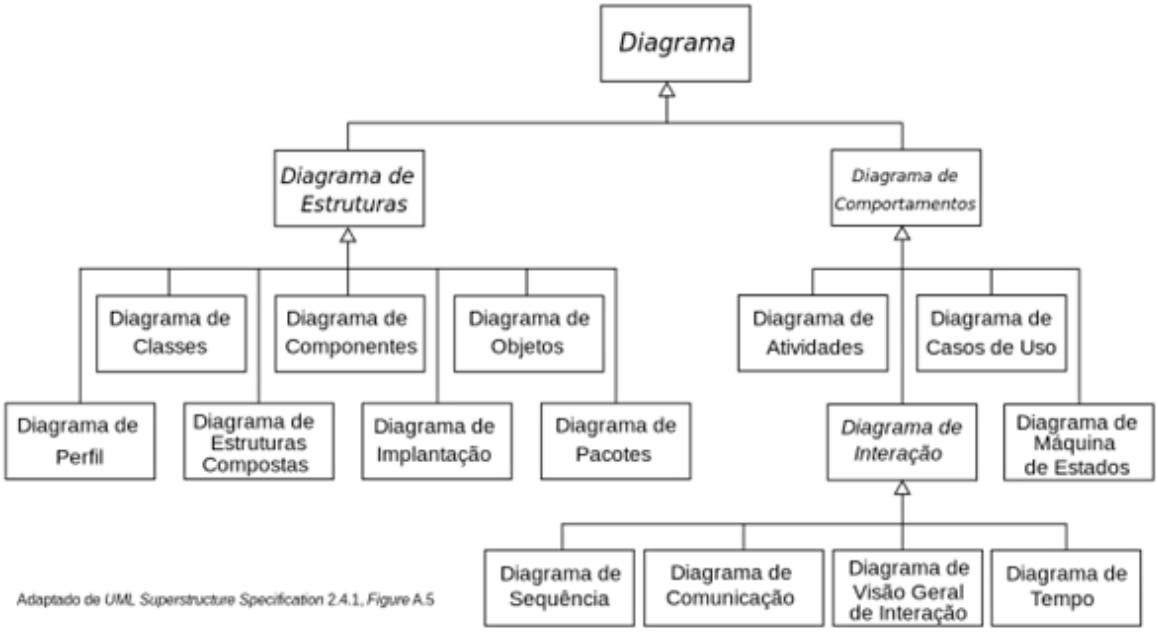
O processo de modelagem envolve a elaboração de dois artefatos básicos: o Diagrama de Casos de Uso e a Descrição de Casos de Uso. Enquanto o Diagrama de Casos de Uso exibe graficamente os atores e seus relacionamentos com as funcionalidades do sistema, a Descrição de Casos de Uso mostra como cada funcionalidade deve funcionar com o sistema em operação.

2. UML

A UML (Linguagem de Modelagem Unificada) nasceu em meados dos anos 1990 com a junção dos métodos de Rumbaugh, Booch e Jacobson, que unificaram o que cada metodologia tinha de melhor para criar uma linguagem única, que seria o padrão para a documentação de sistemas orientado a objetos.

Atualmente a UML se encontra na versão 2.5, e apresenta 14 diagramas divididos entre diagramas estruturais e comportamentais, ambos com 7 tipos de diagramas distintos.

FIGURA 1 **| Composição das UML 2.5**



Fonte: https://pt.wikipedia.org/wiki/UML

A Figura 1 exibe os 14 diagramas da UML 2.5, com a organização em Diagramas de estrutura e comportamentais.

Diagramas Estruturais:

- Classes;
- Componentes;
- Objetos;
- Perfil;
- Estruturas Compostas;
- Implantação;
- Pacotes.

Diagramas Comportamentais:

- Atividades;
- Casos de Uso;
- Máquina de Estados;
- Sequência;
- Comunicação;
- Visão Geral de Interação;
- Tempo.

2.1. DIAGRAMA DE CASOS DE USO

O Diagrama de Casos de Uso é um diagrama visual, que permite ao seu leitor ter uma visão dos requisitos funcionais do sistema, percebendo os relacionamentos de casos de uso com outros casos de uso e entre atores e casos de uso.

Um diagrama de casos de uso, normalmente vai apresentar:

a) Atores: usuários que vão interagir com as funcionalidades do sistema;

b) Casos de Uso: as funcionalidades que o sistema é capaz de executar;

c) Relacionamentos: mostra as interações do sistema e podem ser representadas por:

- Atores com Casos de Uso;
- Casos de Uso com Casos de Uso.

2.2. SOFTWARES DE APOIO

Para elaborar um diagrama de caso de uso, precisamos de um software adequado para isso. Chamamos esta categoria de software normalmente de ferramentas Case (*Computer Aided Software Engineering*).

Essas ferramentas são consideradas Case quando são inteligentes suficientemente para produzir diagramas, economizar tempo com engenharias reversas, e possibilitar a visualização do sistema sob diversos aspectos observáveis.

Para a elaboração de diagramas de caso de uso, indico as seguintes ferramentas:

Lucidchart (https://www.lucidchart.com/pages/pt);

Draw.io (https://app.diagrams.net/);

Astah UML (precisa se registrar como estudante para obter um período de uso grátis https://astah.net/products/free-student-license/).

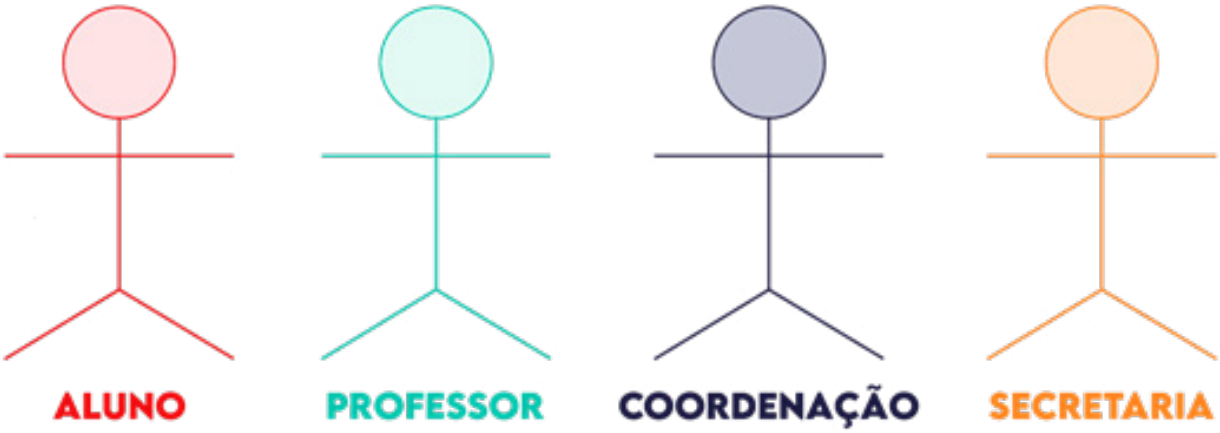
2.3. ATORES

Uma das primeiras tarefas antes de iniciar a elaboração de um diagrama de caso de uso é a identificação dos atores.

Chamamos de ator, aqueles que possuem ao menos uma interação com um requisito do sistema, ou seja, uma funcionalidade que o software será capaz de executar.

Os atores são representados por “bonecos de palito”, ou *stickman*, conforme podemos ver na Figura 2.

FIGURA 2 | **Exemplo de Representação de Atores**



Elaborado pelo autor.

Os atores podem ser pessoas, organizações ou até sistemas que possam interagir com o sistema que esteja sendo modelado, no entanto, atores não podem ser representados por instâncias de pessoas como João, José e Maria, por exemplo.

Exemplos de atores válidos:

- Aluno;
- Professor;
- Secretaria;
- Atendente;
- Usuário;
- Operadora de Cartão de Crédito;
- Sistema de Estoque.

Exemplos de atores inválidos:

- João;
- José;
- Maria.

Para determinarmos um ator, precisamos saber se ele, de fato, vai utilizar o sistema. Um aluno do ensino de alfabetização, por exemplo, não seria um ator, já que ele não seria usuário do sistema da escola, mas sim o seu responsável.

2.4. CASOS DE USO

Os casos de uso já identificados na fase de elicitação de requisitos, agora precisam ser criados no diagrama de caso de uso.

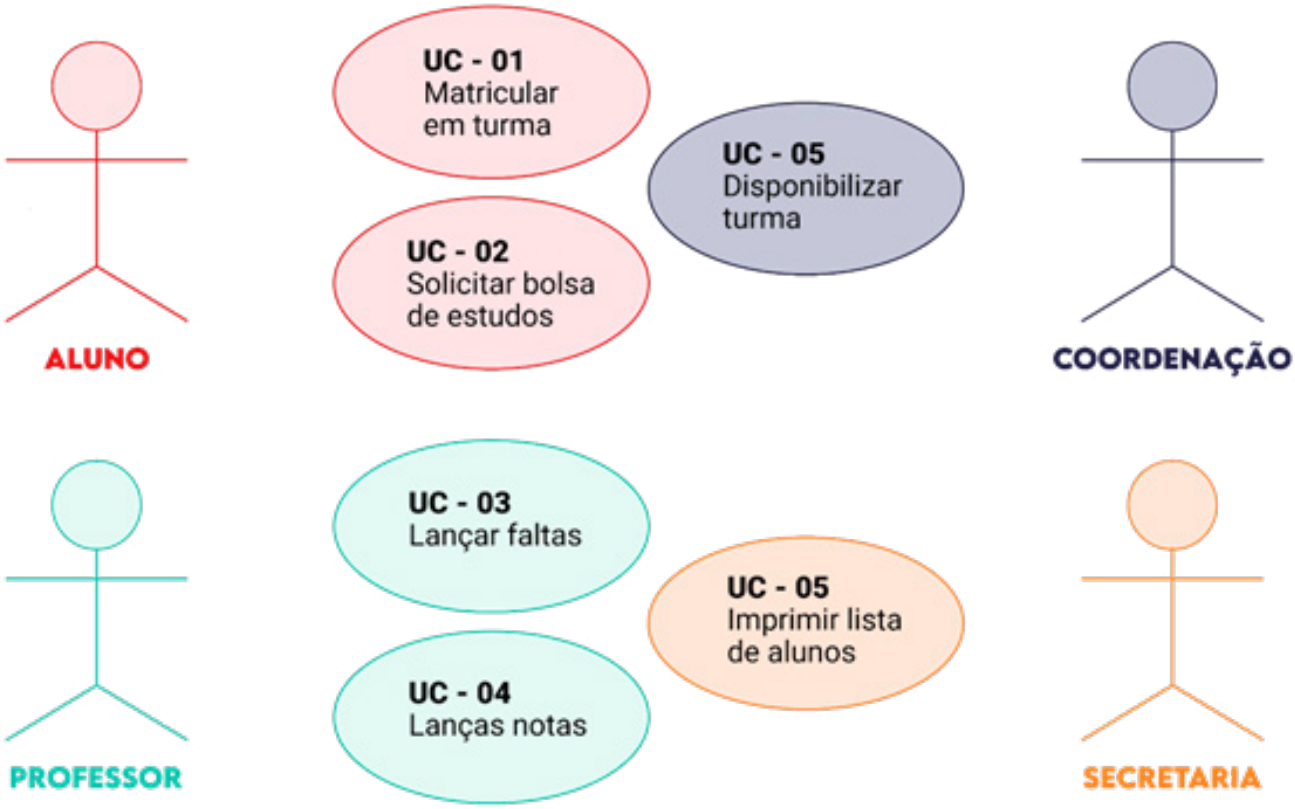
A representação de um caso de uso se dá por uma elipse, com o nome do caso de uso no seu interior. Uma boa prática é identificar os casos de uso com a sigla UC (Use Case) e um número de identificação, por exemplo, UC-01, UC-02, UC-0N.

Para nomear casos de uso usamos um verbo no infinitivo complementado por um predicado que traga o objeto manipulado pelo caso de uso. Exemplos:

- Lançar notas;
- Imprimir Lista de alunos;
- Disponibilizar turma;
- Matricular em turma;
- Lançar faltas;
- Solicitar bolsa de estudos.

A Figura 3 mostra os casos de uso identificados com os seus atores:

FIGURA 3 **| Atores e Casos de Uso**



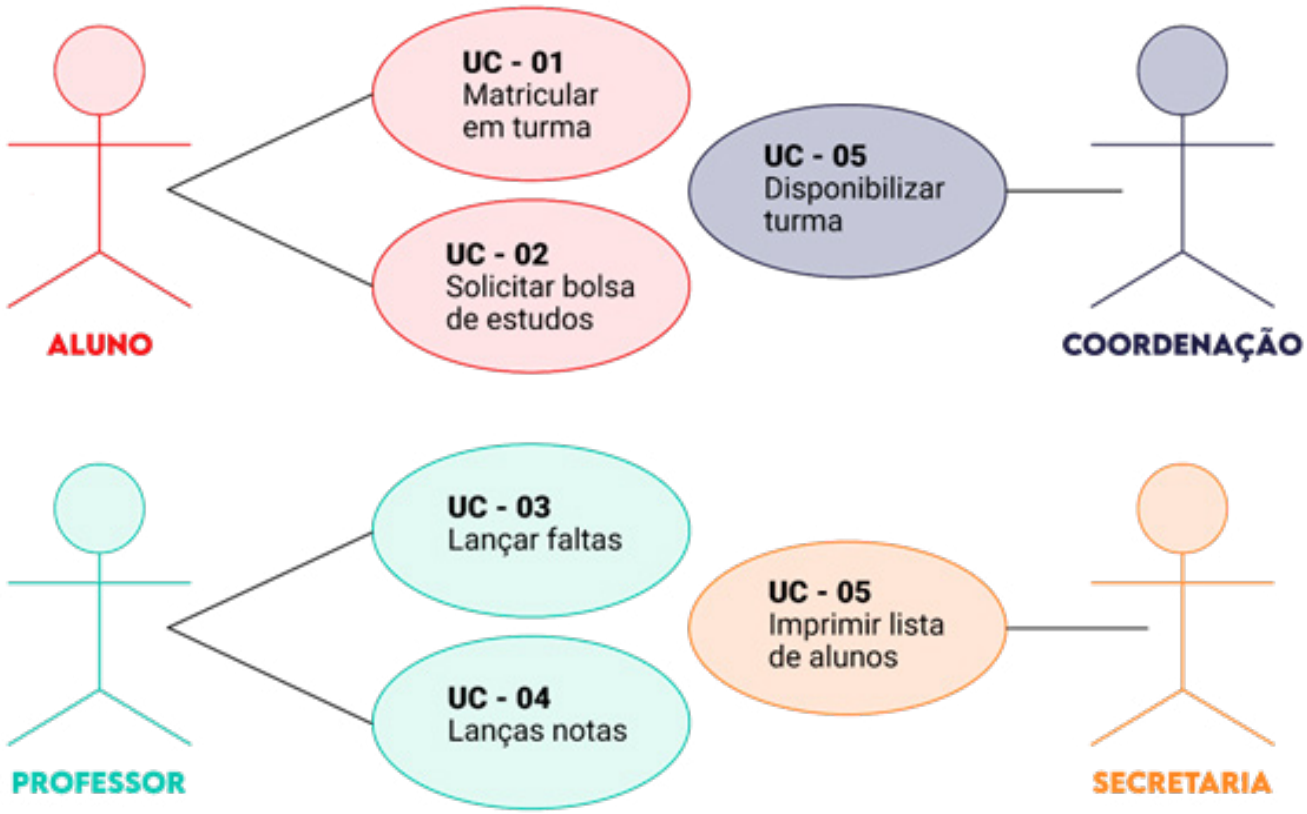
Elaborado pelo autor.

2.5. RELACIONAMENTOS

Agora que já aprendemos a identificar Atores e Casos de Uso, precisamos entender como funcionam os relacionamentos.

Os relacionamentos entre atores indicam quais atores poderão utilizar cada um dos requisitos identificados, e a sua representação se dá através de uma linha reta ligando o ator ao caso de uso. A Figura 4 mostra o relacionamento de nossos atores com os casos de uso que eles podem executar.

FIGURA 4 | **Relacionamento Atores – Casos de Uso**

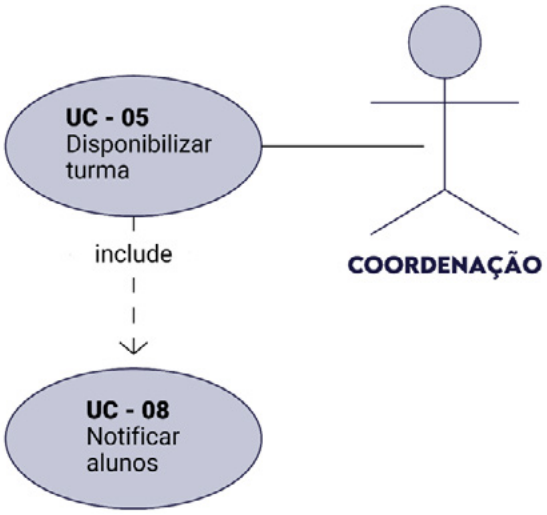


Elaborado pelo autor.

Existem mais um tipo de relacionamento que é usado nesse caso, que é o liga um caso de uso a outro caso de uso. Esse tipo de relacionamento pode ser identificado como um relacionamento de inclusão INCLUDE ou como um relacionamento de extensão EXTEND.

Um relacionamento INCLUDE ocorre quando a execução de um caso de uso implica, obrigatoriamente, na execução de outro. A Figura 5 mostra um exemplo desse relacionamento.

FIGURA 5 **| Relacionamento do Tipo *Include***

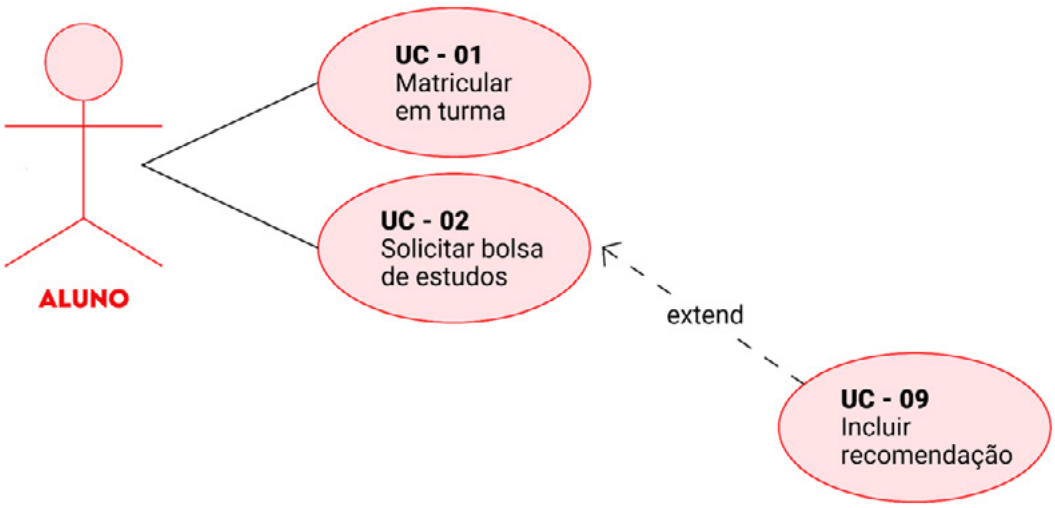


Elaborado pelo autor.

Na Figura 5 vemos um relacionamento de inclusão, que significa que toda vez que o Coordenador disponibiliza uma turma, uma notificação é disparada para os alunos.

Um relacionamento de EXTEND significa que quando um caso de uso é executado, opcionalmente outro caso de uso o pode ser executado. A Figura 6 mostra um exemplo deste tipo de relacionamento.

FIGURA 6 **| Relacionamento do Tipo *Extend***

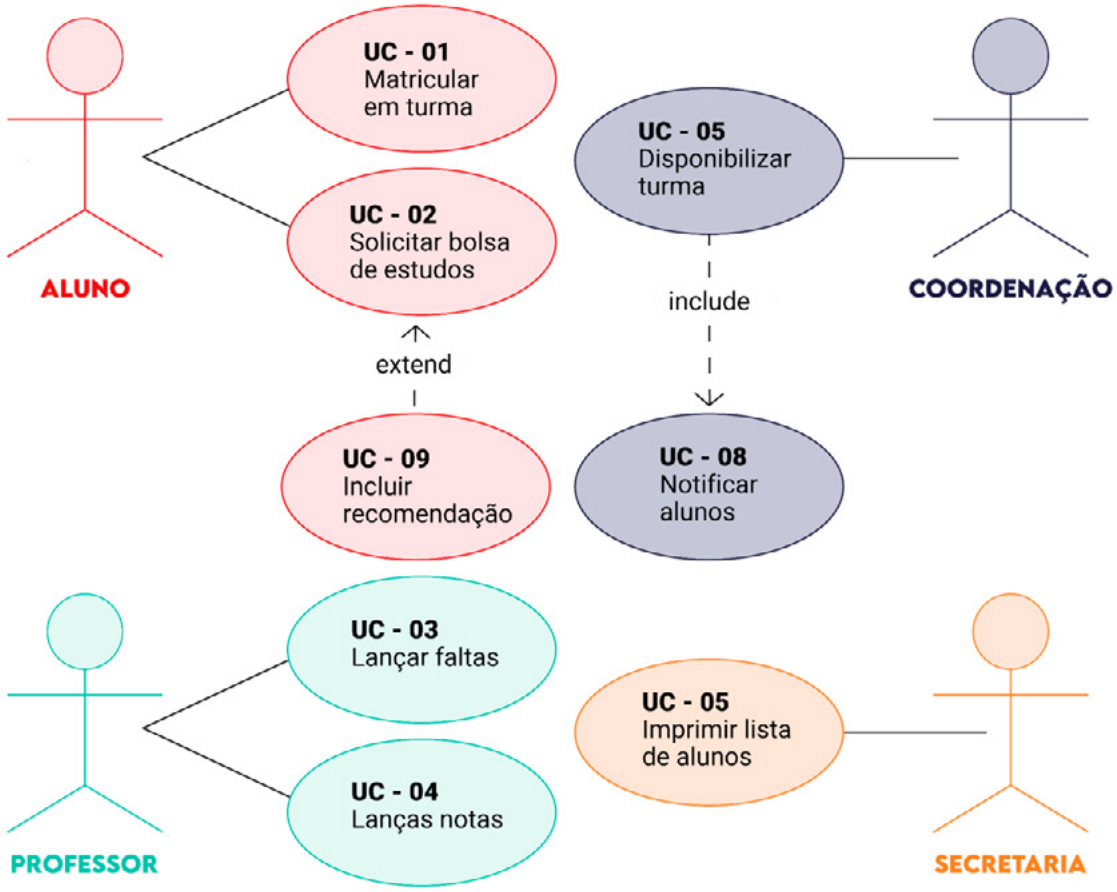


Elaborado pelo autor.

Na Figura 6 vemos um exemplo do relacionamento de extensão. Supondo que ao Solicitar Bolsa de Estudos o aluno pode, opcionalmente, cadastrar uma carta de recomendação do seu coordenador ratificando a importância da concessão da bolsa. Note que nem todos os alunos terão essa carta, mas alguns poderão incluí-la no ato da solicitação.

Realizadas a identificação de atores, casos de uso e relacionamento, temos enfim o Diagrama de Casos de Uso pronto, conforme mostra a Figura 7. Ou seja, modelamos os requisitos identificados na fase anterior.

FIGURA 7 **| Diagrama de Caso de Uso completo**



Elaborado pelo autor.

Após a sua validação, o documento gerado funcionará como um artefato de documentação muito importante para todo o projeto.

CONSIDERAÇÕES FINAIS

Nesta aula aprendemos sobre o processo de modelagem de requisitos utilizando a ferramenta Diagrama de Casos de Uso.

Este tipo de diagrama faz parte de um conjunto de 14 diagramas da UML versão 2.5. A UML é a linguagem de modelagem de dados padrão para a análise orientada a objetos, e possui diversos tipos de visualizações estruturais e comportamentais.

Avançamos de forma importante na especificação alto-nível de requisitos ao aprendermos a utilizar o diagrama de casos de uso, identificando os atores e os relacionando com os casos de uso elicitados na fase anterior.

A partir de exemplos gráficos e práticos, vimos a importância da modelagem e documentação de casos de uso, principalmente pelo fato de que ter a oportunidade de visualizar graficamente os casos de uso torna o entendimento do escopo do sistema muito mais prático e fácil.

MATERIAIS COMPLEMENTARES

Vídeo: *Tutorial de Caso de Uso UML*. Disponível em: https://www.youtube.com/watch?-v=ab6eDdwS3rA. Acesso em: 28 nov. 2022.

Vídeo: *Conhecendo Diagramas de Caso de Uso*. Disponível em: https://www.youtube.com/watch?v=k9hzeGKNoRs. Acesso em: 28 nov. 2022.

Link: *Como escrever requisitos de software de forma simples e garantir o mínimo de erros no sistema e aplicativos*. Disponível em: https://medium.com/lfdev-blog/como-escrever-requisitos-de-software-de-forma-simples-e-garantir-o-m%C3%ADnimo-de-erros-no-sistema-app-74df2ee241cc. Acesso em: 28 nov. 2022.

Link: *7 Ferramentas gratuitas para criar diagramas de caso de uso*. Disponível em: https://www.profissionaisti.com.br/7-ferramentas-online-gratuitas-para-criar-diagramas-uml/. Acesso em: 28 nov. 2022.

Link: *O que é UML e Diagramas de Caso de Uso: Introdução Práticas à UML*. Disponível em: https://www.devmedia.com.br/o-que-e-uml-e-diagramas-de-caso-de-uso-introducao-pratica-a-uml/23408. Acesso em: 28 nov. 2022.

REFERÊNCIAS

PRESSMAN, R.G. *Engenharia de Software.* 9ª ed. Rio de Janeiro: McGraw-Hill, 2021.

SOMMERVILLE, I. *Engenharia de Software.* 10ª ed. São Paulo: Pearson Addison. Wesley, 2019.

AULA 5 – ESPECIFICAÇÃO E DOCUMENTAÇÃO DE REQUISITOS

OBJETIVO DA AULA

Conhecer as boas práticas de especificações de casos de uso, indicando a maneira correta de documentar os requisitos identificados durante a elicitação.

APRESENTAÇÃO

Esta aula tem o objetivo de apresentar o processo de documentação de casos de uso. Esta documentação permitirá que o analista possa descrever o passo a passo da execução do caso de uso.

Esta documentação de caso de uso faz parte da finalização do processo de elicitação, documentação e validação de requisitos, que estamos estudando nesta unidade.

Nesta aula será mostrado o processo de documentação de casos de uso e também apresentará as boas práticas necessárias para criar bons documentos e, assim, apoiar os processos futuros, como o da implementação do sistema.

1. ESPECIFICAÇÃO DE CASOS DE USO

Conforme estamos acompanhando nesta unidade, o processo de elicitação de requisitos é composto de diversas etapas. Já falamos sobre as técnicas que podem ser utilizadas na elicitação de requisitos, já abordamos a construção de casos de uso para a modelagem dos casos de uso, e agora vamos avançar para a especificação e documentação dos casos de uso.

Ao criar o diagrama de casos de uso, o próximo passo é documentá-los através de suas especificações. E que seria especificação?

Especificar casos de uso significa descrevê-los, informando o passo a passo de sua execução, a partir de um fluxo de dados iniciado pelo usuário e continuado pelo próprio sistema.

As especificações são bastante úteis para que o desenvolvedor possa entender como determinada funcionalidade deverá ser executada no sistema. Uma boa descrição de caso de uso será um verdadeiro trunfo nas mãos dos desenvolvedores quando estes iniciarem a programação do software.

2. MODELOS DE DOCUMENTAÇÃO DE CASOS DE USO

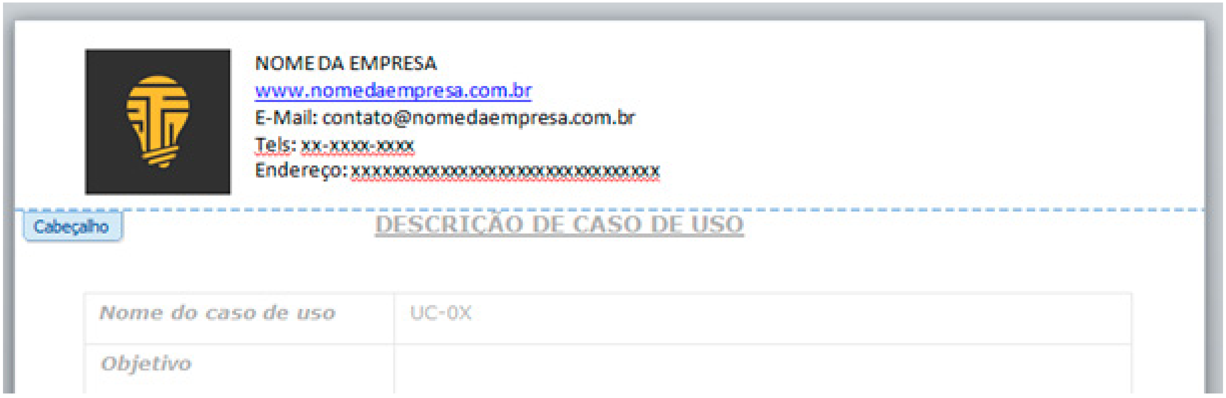
A especificação ou documentação de caso de uso é normalmente feita em um editor de texto comum, embora haja ferramentas Case que possibilitam ao analista descrever os casos de uso dentro dela.

Antes de mais nada, considero uma boa prática criar um *template* próprio para esse fim. A elaboração de um *template* padroniza as descrições dos casos de uso, além de ajudar na organização de todo o projeto.

Comece criando cabeçalho e rodapé com os dados importantes da organização, como nome, telefones, e-mail, endereço e logotipo. Lembre-se que estou apenas dando dicas de como obter melhores resultados em seu projeto, mas cada profissional e organização deve se sentir à vontade para tratar essa questão da maneira que julgarem melhor.

A Figura 1 exibe um exemplo de cabeçalho:

FIGURA 1 | **Cabeçalho**



Fonte: Elaborado pelo autor.

A partir daí é importante definir quais são as informações que estarão no *template* da descrição do caso de uso.

As informações mais importantes que encontramos nas especificações de caso de uso são:

- Nome e identificação do Caso de Uso;
- Objetivo;
- Ator Primário;
- Atores Secundários;
- Pré-Condições;
- Fluxo Principal;
- Fluxo Alternativo;
- Pós-Condições;
- Fluxo de Exceção;
- Regras de Negócio;
- Requisitos Não Funcionais;

- Autor do Documento;
- Data da Elaboração;
- Versão do Documento.

Agora vamos verificar na Tabela 1 como preenchemos cada elemento do diagrama:

TABELA 1 | **Elementos de um Documento de Caso de Uso**

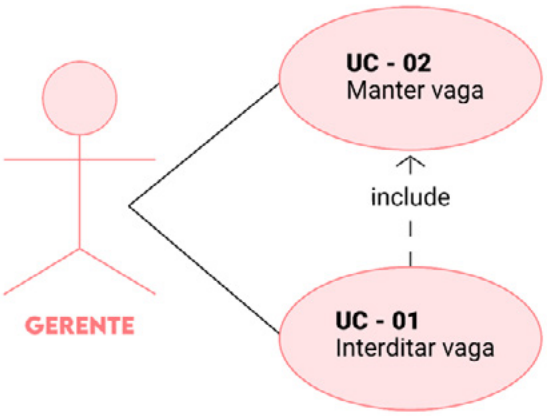
Elemento	Objetivo no Documento
Nome e identificação do Caso de Uso	O caso de uso deve ser identificado. Uma boa prática é criar um código para ele como, por exemplo, UC-01
Objetivo	Deve ser descrito, de maneira simples e direta, o objetivo do caso de uso
Atores Primários	Devem ser listados os atores que podem executar o caso de uso
Atores Secundários	Devem ser listados os atores que indiretamente tem o seu trabalho afetado pela execução do caso de uso
Pré-Condições	Condições que devem ser satisfeitas para que um caso de uso seja executado
Fluxo Principal	Devem ser descritas as interações usuário x sistema para a execução do caso de uso
Fluxo Alternativo	Devem ser descritos os fluxos em que o processo desviou do caminho normal de execução do fluxo principal
Pós-Condições	Deve descrever o resultado final obtido após a execução do caso de uso
Fluxo de Exceção	Devem ser assinaladas as situações em que erros acontecem impedindo a execução do caso de uso
Regras de Negócio	Deve documentar as regras de negócio do domínio
Requisitos Não Funcionais	Devem ser listados os requisitos não funcionais relacionados ao caso de uso
Autor do Documento	Nome dos autores que elaboraram o documento
Data da Elaboração	Data de Elaboração ou atualização do documento
Versão do Documento	Este campo deve ser usado para o versionamento do documento

Elaborado pelo autor.

Embora haja várias opções de itens a serem preenchidos, é normal acharmos documentação de caso de uso elaboradas com menos ou mais itens. O analista pode escolher aqueles que julgarem mais importante, não esquecendo, porém, dos principais como nome, objetivo, fluxo principal, fluxo alternativo (se houver), pré e pós-condições.

A título de ilustração desta aula, vou apresentar um exemplo completo de um caso de uso preenchido. Para isso, vamos utilizar o trecho do diagrama de caso de uso da Figura 2.

FIGURA 2 **| Trecho de um Diagrama de Caso de Uso**



Elaborado pelo autor.

Neste diagrama de um hipotético estacionamento, estamos documentando que:

- O ator Gerente pode Interditar uma Vaga;
- O ator Gerente pode Manter uma Vaga;
- Quando o Caso de Uso Interditar Vaga é executado, automaticamente o Caso de Uso Manter Vaga também é executado.

Vamos então descrever os dois casos de uso, iniciando pelo caso de uso Interditar Vaga. Nele queremos representar uma operação no sistema em que o gerente pode desabilitar a vaga para que ela esteja indisponível enquanto estiver em manutenção.

A Tabela 2 mostra a descrição completa do caso de uso. Note principalmente o Fluxo Principal, que deve ser representado a partir da interação usuário x sistema, sempre iniciada pelo usuário.

TABELA 2 **| Descrição Completa de Caso de Uso**

Nome do caso de uso	UC01 – Interditar vaga
Objetivo	Este caso de uso tem o objetivo de interditar uma vaga por motivos de impedimento de uso, como obras e vazamentos
Ator primário	Gerente
Atores secundários	Não há
Precondições	A vaga tem que estar cadastrada no sistema A vaga tem que estar com o status livre

Fluxo Principal	1) Gerente solicita interdição da vaga 2) Sistema apresenta lista de vagas disponíveis 3) Gerente escolhe vaga a ser interditada 4) Sistema solicita o motivo da interdição da vaga 5) Gerente insere o motivo da interdição da vaga 6) Sistema solicita confirmação da interdição da vaga 7) Gerente confirma interdição da vaga 8) Sistema interdita a vaga, atualizando o cadastro da vaga (UC-02) e informa ao usuário que a interdição foi bem-sucedida 9) Fim do caso de uso
Fluxo Alternativo	(3) a) A vaga a ser interditada está em uso b) Gerente finaliza caso de uso
Fluxo Exceção	(2) a) O número de vagas para interdição foi excedido b) Sistema informa ao cliente que o número de vagas para interdição foi excedido c) Gerente finaliza o caso de uso
Pós-condições	Vaga interditada
Regras de negócio	RN01 – O sistema não deve autorizar a interdição de mais de 50% das vagas
Pontos de Inclusão	UC-02 – Manter Vaga
Requisitos não funcionais	Não há
Autor	Anderson Nascimento
Data de elaboração	29/11/2022
Versão	1.0

Elaborado pelo autor.

Um caso de uso do tipo “Manter” significa que as seguintes operações podem ser executadas:

- Inclusão;
- Leitura;
- Atualização;
- Deleção.

Ou seja, os casos de uso “Manter” possibilitam as operações conhecidas como CRUD (*Create*, *Read*, *Update* e *Delete*) e são representados de maneira um pouco diferente.

Teremos um caso de uso principal, e quatro casos de uso representando os subfluxos Cadastrar, Alterar, Remover e Consultar.

A Tabela 3 mostra a descrição do caso de uso principal.

TABELA 3 | **Descrição do Caso de Uso Principal**

Nome do caso de uso	UC-02 – Manter Vaga
Objetivo	Este caso de uso tem o objetivo de cadastrar, alterar, consultar ou excluir uma vaga
Ator primário	Gerente
Atores secundários	Não há
Precondições	Não há
Fluxo Principal	1) Gerente inicia a manutenção de uma vaga 2) Sistema apresenta uma lista com as opções cadastrar, alterar, consultar e excluir uma vaga 3) De acordo com o tipo de manutenção escolhido pelo Gerente, um dos subfluxos é executado: a) Se o Gerente deseja incluir uma nova Vaga, o subfluxo Cadastrar Vaga é executado b) Se o Gerente deseja Alterar informações de uma Vaga já cadastrada, o subfluxo Alterar Vaga é executado c) Se o Gerente deseja excluir uma Vaga já cadastrada, o subfluxo Excluir Vaga é executado d) Se o Gerente deseja consultar informações sobre uma ou mais Vagas cadastradas, o subfluxo Consultar Vaga é executado
Fluxo Alternativo	Não há
Fluxo Exceção	Não há
Pós-condições	Manutenção efetivada
Regras de negócio	Não há
Requisitos não funcionais	Não há
Autor	Anderson Nascimento
Data	29/11/2022

Fonte: Elaborado pelo autor.

Na sequência, as tabelas 4, 5, 6 e 7 mostram respectivamente os subfluxos executados para Cadastrar Vaga, Alterar Vaga, Remover Vaga e Consultar Vaga.

TABELA 4 | **Subfluxo Cadastrar Vaga**

Subfluxo Cadastrar Vaga	1) O sistema solicita ao Gerente o preenchimento dos seguintes dados: código, status, largura, comprimento, categoria, coberta, localização e observações 2) O Gerente preenche os dados solicitados e confirma a inclusão 3) O sistema realiza a inclusão dos dados informados pelo Gerente 4) O sistema exibe uma mensagem informando que a inclusão da Vaga foi efetivada com sucesso 5) Fim do subfluxo Cadastrar Vaga

Autor	Anderson Nascimento
Data	29/11/2022

Fonte: Elaborado pelo autor.

TABELA 5 **| Subfluxo Alterar Vaga**

Subfluxo Alterar Vaga	1) O sistema solicita a identificação da vaga 2) Gerente informa a vaga a ser alterada 3) Sistema lista os dados da vaga informada 4) Gerente altera o(s) dado(s) desejado(s) da vaga informada e confirma alteração 5) Sistema solicita confirmação de alteração dos dados da vaga 6) Gerente confirma alteração dos dados da vaga 7) Sistema informa que a alteração foi realizada 8) Fim do subfluxo Alterar vaga
Autor	Anderson Nascimento
Data	29/11/2022

Fonte: Elaborado pelo autor.

TABELA 6 **| Subfluxo Remover Vaga**

Subfluxo Remover Vaga	1) O Sistema solicita a identificação da vaga a ser removida 2) Gerente informa a vaga a ser removida 3) Sistema exibe as informações da vaga a ser removida e solicita a confirmação de remoção 4) Gerente confirma a remoção da vaga 5) Sistema informa que a remoção foi realizada 6) Fim do subfluxo Alterar vaga
Fluxo Alternativo	(3) a) Sistema informa que a vaga informada para remoção está ocupada naquele momento b) Retornar ao passo 2 do Subfluxo Remover Vaga
Autor	Anderson Nascimento
Data	29/11/2022

Fonte: Elaborado pelo autor.

TABELA 7 **| Subfluxo Consultar Vaga**

Subfluxo Consultar Vaga	1) O Sistema solicita a identificação da vaga a ser consultada 2) Gerente informa a vaga a ser consultada 3) Sistema exibe as informações da vaga a ser consultada 4) Fim do subfluxo Consultar vaga
Autor	Anderson Nascimento
Data	29/11/2022

Fonte: Elaborado pelo autor.

A descrição de caso de uso pode ser feita de várias maneiras diferentes, não existindo então apenas uma forma de realizá-la. O mais importante neste caso é que o documento não apresente inconsistência, e que possa apoiar o processo de análise e desenvolvimento do sistema.

CONSIDERAÇÕES FINAIS

Nesta aula abordamos a documentação de requisitos a partir das descrições de casos de uso.

Foi apresentando um modelo de documento que pode ser utilizado para especificação do requisito, trazendo os principais itens que ajudam a moldar as características em um requisito.

Vimos um exemplo de descrição de caso de uso simples e também um exemplo de descrição de caso de uso do tipo “manter”. Este tipo de caso de uso é frequentemente utilizado em qualquer descrição de sistema, pois apresenta as operações básicas de inclusão, alteração, remoção e consulta, mais popularmente conhecido como CRUD.

Por fim, vale a pena ressaltar a importância da especificação de requisitos, no intuito de apoiar o processo do desenvolvimento do software.

MATERIAIS COMPLEMENTARES

Link: *Definindo o Escopo: Modelos de Caso de Uso*. Disponível em: http://www.lyfreitas.com.br/ant/pdf/Modelo_de_Caso_de_Uso.pdf. Acesso em: 29 nov. 2022.

Link: *Modelagem de Casos de Uso*. Disponível em: https://www.macoratti.net/11/10/uml_rev1.htm. Acesso em: 29 nov. 2022.

REFERÊNCIAS

PRESSMAN, R.G. *Engenharia de Software.* 9ª ed. Rio de Janeiro: McGraw-Hill, 2021.

SOMMERVILLE, I. *Engenharia de Software.* 10ª ed. São Paulo: Pearson Addison. Wesley, 2019.