Arquitetura de Software e Engenharia de Requisitos

Brasília-DF.

Elaboração

Jorge Umberto Scatolin Marques

Produção

Equipe Técnica de Avaliação, Revisão Linguística e Editoração

Sumário

Apresentação

Caro aluno

A proposta editorial deste Caderno de Estudos e Pesquisa reúne elementos que se entendem necessários para o desenvolvimento do estudo com segurança e qualidade. Caracteriza-se pela atualidade, dinâmica e pertinência de seu conteúdo, bem como pela interatividade e modernidade de sua estrutura formal, adequadas à metodologia da Educação a Distância – EaD.

Pretende-se, com este material, levá-lo à reflexão e à compreensão da pluralidade dos conhecimentos a serem oferecidos, possibilitando-lhe ampliar conceitos específicos da área e atuar de forma competente e conscienciosa, como convém ao profissional que busca a formação continuada para vencer os desafios que a evolução científico-tecnológica impõe ao mundo contemporâneo.

Elaborou-se a presente publicação com a intenção de torná-la subsídio valioso, de modo a facilitar sua caminhada na trajetória a ser percorrida tanto na vida pessoal quanto na profissional. Utilize-a como instrumento para seu sucesso na carreira.

Conselho Editorial

Organização do Caderno de Estudos e Pesquisa

Para facilitar seu estudo, os conteúdos são organizados em unidades, subdivididas em capítulos, de forma didática, objetiva e coerente. Eles serão abordados por meio de textos básicos, com questões para reflexão, entre outros recursos editoriais que visam tornar sua leitura mais agradável. Ao final, serão indicadas, também, fontes de consulta para aprofundar seus estudos com leituras e pesquisas complementares.

A seguir, apresentamos uma breve descrição dos ícones utilizados na organização dos Cadernos de Estudos e Pesquisa.

Provocação

Textos que buscam instigar o aluno a refletir sobre determinado assunto antes mesmo de iniciar sua leitura ou após algum trecho pertinente para o autor conteudista.

Para refletir

Questões inseridas no decorrer do estudo a fim de que o aluno faça uma pausa e reflita sobre o conteúdo estudado ou temas que o ajudem em seu raciocínio. É importante que ele verifique seus conhecimentos, suas experiências e seus sentimentos. As reflexões são o ponto de partida para a construção de suas conclusões.

Sugestão de estudo complementar

Sugestões de leituras adicionais, filmes e sites para aprofundamento do estudo, discussões em fóruns ou encontros presenciais quando for o caso.

Atenção

Chamadas para alertar detalhes/tópicos importantes que contribuam para a síntese/conclusão do assunto abordado.

Saiba mais

Informações complementares para elucidar a construção das sínteses/conclusões sobre o assunto abordado.

Sintetizando

Trecho que busca resumir informações relevantes do conteúdo, facilitando o entendimento pelo aluno sobre trechos mais complexos.

Para (não) finalizar

Texto integrador, ao final do módulo, que motiva o aluno a continuar a aprendizagem ou estimula ponderações complementares sobre o módulo estudado.

Introdução

O desenvolvimento de software possui técnicas diversas que auxiliam os desenvolvedores, arquitetos, engenheiros e clientes a se entenderem para elaborar uma ideia e colocá-la no mundo digital. As técnicas variam desde encontros, protótipos, entrevistas, diagramas etc. Além de possuir muitas técnicas, muitas fases também são utilizadas para que todos atinjam seus objetivos. Uma delas é a Engenharia de requisitos.

Não é apenas por ser a primeira fase do ciclo de vida de um desenvolvimento do software que a Engenharia de Requisitos é a mais importante, mas, é o fato de o software já ser funcional logo nessa etapa.

Quando dissemos que o software precisa “praticamente funcionar” nessa fase, isso se deve pelo fato que é melhor que as coisas se ajustam nessa etapa do que quando o software já estiver em funcionamento em produção. Descobrir uma falha de requisitos com todos os usuários utilizando o sistema gera um custo muito alto para a empresa de desenvolvimento e a empresa que contratou os serviços.

Nessa fase, deverá haver um conhecimento muito grande entre as partes interessadas para poder chegar ao objetivo. Normalmente, de forma mais genérica, atingir o objetivo é simplesmente fazer o software funcionar conforme os requisitos. Excluindo a simplicidade da frase, fazer o software funcionar conforme os requisitos vai requerer muitos testes nos requisitos, muitas validações, muitas reuniões, muitas discussões.

Portanto, a Engenharia de Requisitos, como será demonstrado nesse material, dará todo o suporte para o ciclo de vida do desenvolvimento do software, desde sua concepção, passando pelas alterações de requisitos, até seu pleno funcionamento e suporte.

Objetivos

» Compreender as melhores práticas de gerenciamento de requisitos.

» Estudar as principais técnicas e boas práticas de Engenharia de Requisitos.

» Entender técnicas para elaborar um produto de valor ao cliente.

CONCEITOS DE REQUISITOS

UNIDADE I

CAPÍTULO 1

Definição de requisitos

Quando vamos instalar um software, a maioria dos fabricantes nos disponibiliza as especificações de requisitos. Essas especificações são informações do ambiente mínimo necessário para que esse software possa funcionar. No processo de desenvolvimento de software, também é utilizada a palavra requisitos, entretanto, seu significado pode necessitar de um entendimento mais detalhado.

Quando vamos desenvolver um software, temos que saber muitas características além de seu funcionamento. Nesse processo, quando a equipe já está mais madura, atenta-se a características relacionadas diretamente ao processo de produção, como linguagem de programação, tipo de arquitetura, gerenciador de banco de dados, pessoas interessadas, domínio do negócio, planejamento do projeto e somente depois que todos os requisitos são apresentados.

Quando nos é apresentada a ideia da concepção de um software, podemos estar nos deparando com um novo produto ou manutenção já existentes. Um novo software pode contemplar desde a criação "do zero[1]", acrescentar novos módulos, ou ainda, acrescentar tantas novas funcionalidades que acaba se transformando em um novo produto. A manutenção de software já visa fazer correções, atualizações em módulos de códigos com o propósito da aplicação não fica em desuso.

Não importa se iremos desenvolver um novo, atualizações ou manutenções. O importante é saber que, para que o processo de codificação seja iniciado, é necessário sabermos, de antemão, os requisitos do que iremos fazer.

Segundo Pfleeger (2004, p. 111), um requisito é uma característica ou descrição de algo que um novo sistema ou de uma atualização, são capazes de realizar, para atingir seus objetivos.

Já Sommervile (2001, p. 57) menciona que os requisitos de um sistema são as descrições das funcionalidades oferecidas que o sistema deverá executar e,, também, as restrições a seu funcionamento. Os requisitos refletem as necessidades dos clientes para um sistema que serve a um propósito, como controlar um equipamento, colocar um pedido ou encontrar informações. O processo de descobrir, analisar, documentar e verificar esses serviços e restrições é chamado Engenharia de Requisitos (RE, do inglês *requirements engineering*).

Um requisito é definido da seguinte forma:

» Uma condição ou capacidade necessária ao usuário para resolver um problema ou atingir um objetivo.

» Uma condição ou capacidade que deve ser atendida ou possuída por um sistema ou componentes de sistema para satisfazer um contrato, norma, especificação ou outro documento formalmente imposto.

» Uma representação documentada de uma condição ou capacidade como em 1 e 2.

Fernandes (2017, p. 66) cita o IEEE:

» A necessidade ou a condição que alguém precisa para resolver um problema ou atingir um objetivo.

» Uma condição ou capacidade a ser verificada para atender a um contrato, uma norma, uma especificação, ou alguma outra formalidade.

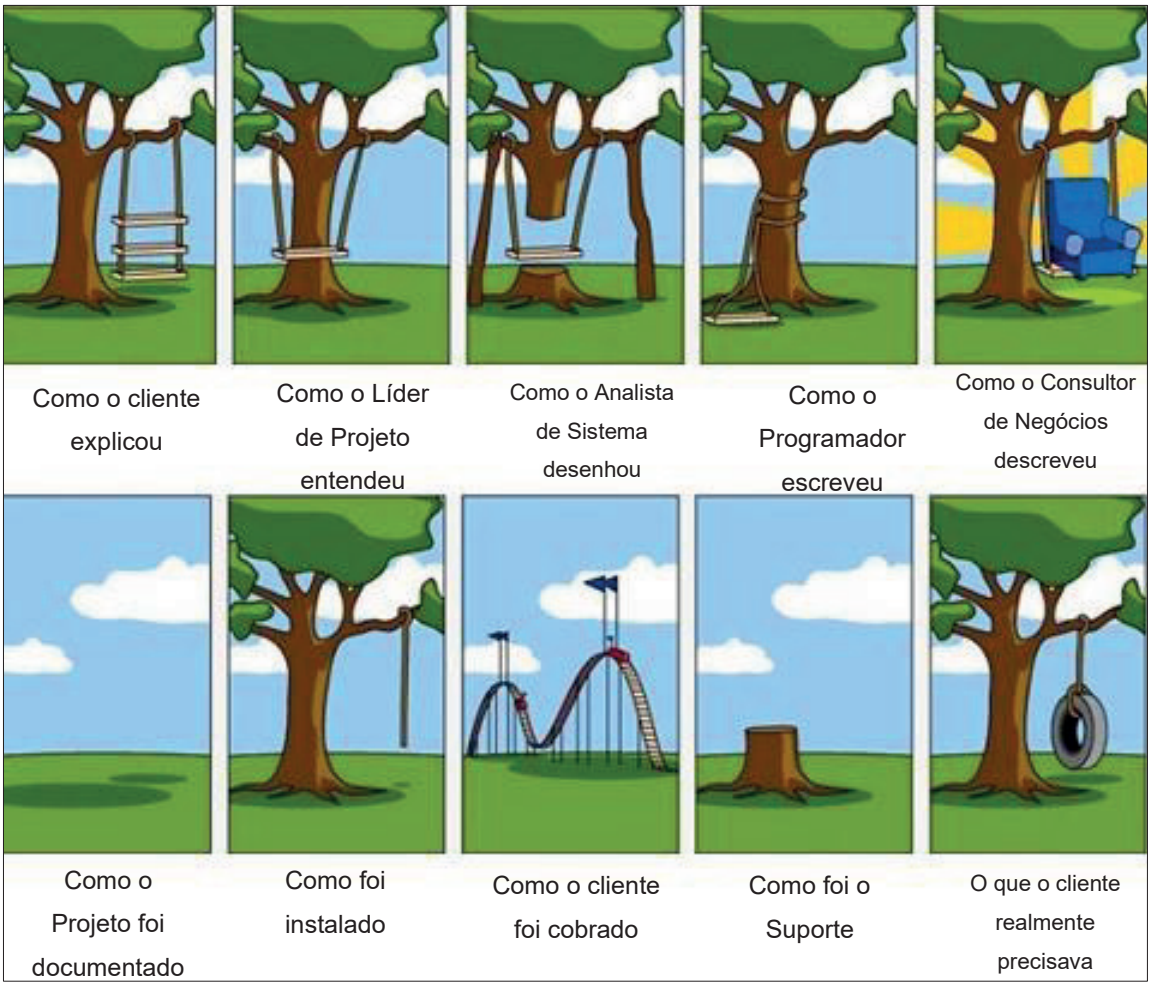
» Uma representação de uma condição ou de uma capacidade documentada.

Os objetivos que o projeto deverá atingir serão descritos em documentos específicos, denominados “Documentos de Requisitos”, ricos em detalhes, desde funcionalidades, pessoas interessadas, cronogramas, atividades, definições de entregas etc.

Os requisitos bem documentados são fundamentais para o desenvolvimento do software. Consequentemente, um requisito mal documentado ou não compreendido por toda equipe pode por a perder todo o projeto. Os requisitos de qualquer projeto de desenvolvimento de software precisam ser muito bem elaborados e claramente entendidos por todos os envolvidos, mas talvez o mais importante seja que eles não sejam descartados ou comprometidos na metade do projeto.

Segundo Pfleeger (2004, p. 112), é muito comum projetos começarem com os ânimos bem intensos para elaborar todas as características de uma boa análise de requisitos, mas, conforme o tempo passa, esses mesmos ânimos vão perdendo o fôlego. Isso acontece pois muitos dos envolvidos são desenvolvedores e estão muito ansiosos para iniciar o processo de codificação. Por isso, o processo de identificação dos requisitos é muito importante.

Figura 1. Exemplo de requisitos mal elaborados.



Fonte: adaptado de: https://miro.medium.com/max/1146/1*ApoalZicyU0b7Jgg9MMJhw.jpeg. Acesso em: 23/1/2020.

As pessoas, sejam os gerentes ou qualquer outro leitor dos **requisitos de usuário**, não se preocupam com a forma de como o sistema será implementado ou não estão interessados nos recursos detalhados do sistema. Já os leitores dos r**equisitos de sistema** precisam saber mais detalhadamente o que o sistema fará, porque estão interessados em como ele apoiará os processos dos negócios ou porque estão envolvidos na implementação do sistema.

O "não-técnico" significa que os responsáveis por extrair informações do cliente, pertinente ao assunto, não seja feita de relacionada a alguma linguagem de programação ou mesmo uma linguagem mais simbólica, como a UML, por exemplo. O cliente também deverá informar aos requisitantes informações "não-técnicas" de seu segmento. Normalmente, isso ocorre quando algum software específico de uma determinada área que não esteja ligada a gestão de empresas, que o software mais utilizado em todo mundo. Um bom exemplo é o desenvolvimento de softwares para área de genética, manufaturas químicas etc.

Em algum determinado momento, a equipe de analistas de requisitos deverá dominar totalmente o assunto-alvo. Porém, como foi mencionado, não nos primeiros contatos com o cliente.

Para colher essas informações, podemos utilizar questionários, demonstrações de sistemas do mesmo segmento e, até mesmo, desenvolvermos alguns protótipos para melhor entendimento. Os requisitos, então, demonstrados de forma que a equipe de desenvolvimento e o cliente possam concordar sobre o que o sistema irá fazer, ou seja, suas principais funcionalidades.

Em seguida, esses requisitos poderão ser transformados em uma representação, de tal maneira que os projetistas possam convertê-lo em um projeto de sistemas.

Segundo Pfleeger (2004, p. 112), passar por uma etapa de **verificação** garante que os requisitos estejam completos, corretos e consistentes, e uma etapa de **validação** garante o que o cliente pretende ver no produto final.

Os requisitos de software são frequentemente classificados como requisitos funcionais e requisitos não funcionais:

- **Requisitos funcionais**: são declarações de funcionalidade que o sistema deve fornecer, de como o sistema deve reagir a entradas específicas e de como o sistema deve se comportar em determinadas situações. Em alguns casos, os requisitos funcionais também podem explicitar o que o sistema não deve fazer.

» **Requisitos não funcionais**: são restrições aos requisitos ou funções oferecidos pelo sistema. Incluem restrições de timing, restrições no processo de desenvolvimento e restrições impostas pelas normas. Ao contrário das características individuais ou serviços do sistema, os requisitos não funcionais, muitas vezes, aplicam-se ao sistema como um todo.

» **Requisitos do domínio**: derivam do domínio da aplicação e refletem suas características e podem ser tanto requisitos funcionais ou requisitos não funcionais. A diferença entre eles é que requisitos de domínio normalmente abrangerá conhecimento fora dos limites de qualquer sistema de software (SOMMERVILLE, 2007).

Os requisitos de domínio podem ser expressos usando terminologia de domínio especializada ou referência a conceitos de domínio. Como esses requisitos são especializados, os engenheiros de software geralmente acham difícil entender como eles estão relacionados a outros requisitos do sistema.

Os requisitos de domínio são importantes porque geralmente refletem os fundamentos do domínio do aplicativo. Se esses requisitos não forem atendidos, pode ser impossível fazer o sistema funcionar satisfatoriamente. Por exemplo, os requisitos para o sistema de Nota Fiscal Eletrônica, que calcula os impostos segundo determinadas regras de tributação, incluem seguintes requisitos de domínio:

› CFOP 6929 – lançamento efetuado em decorrência de emissão de documento fiscal relativo a operação ou prestação também registrada em equipamento Emissor de Cupom Fiscal – ECF.

Regra de Negócios ou Restrições: são restrições aplicadas a uma operação ligadas às operações comerciais de uma empresa, de forma que atenda ao negócio, conforme uma lista de regras estabelecidas. Normalmente, as regras de negócio já estão definidas antes da necessidade da concepção de um sistema. É muito importante a comparação do fluxo de processo, do produto que está sendo desenvolvido, com as regras de negócios da empresa. Ela pode acompanhar esse fluxo, melhorá-lo, mas nunca o confrontar. Se isso acontecer, será necessário descobrir uma forma de o negócio do cliente não ser prejudicado.

WEIGERS, K. M. **Software Requirements**. 2. ed. Microsoft Press, 2003. Esse livro, projetado para escritores e usuários de requisitos, discute boas práticas de engenharia de requisitos.

SOMMERVILLE, I. **Integrated requirements engineering**: a tutorial. IEEE Software, v. 22, n. 1, jan.-fev. 2005. Disponível em: http://dx.doi.org/10.1109/MS.2005.13.

Esse é um artigo tutorial que escrevi. Nele, discuto as atividades da engenharia de requisitos e como elas podem ser adaptadas para as práticas modernas da engenharia de software. **Mastering the Requirements Process, 2nd edition.** Um livro bem escrito e fácil de ler, que se baseia em um método específico (VOLERE), mas que também inclui bons conselhos gerais sobre engenharia de requisitos. (ROBERTSON, S.; ROBERTSON, J. Mastering the Requirements Process. 2. ed. Addison-Wesley, 2006.)

CHENG, B. H. C.; ATLEE, J. M. **Research Directions in Requirements Engineering**. Proc. Conf on Future of Software Engineering, IEEE Computer Society, 2007. Disponível em: http://dx.doi.org/10.1109/FOSE.2007.17.

Esse é um bom levantamento da pesquisa de engenharia de requisitos, que destaca os desafios das futuras pesquisas da área para resolver questões como escala e agilidade.

CAPÍTULO 2
Requisitos funcionais

Segundo Sommerville (2011, p. 59), os requisitos funcionais de um sistema descrevem o que ele deve fazer, também descrevem a interação entre o sistema e seu ambiente, como o sistema deve se comportar quando acionado por certos estímulos.

Os requisitos funcionais de software ajudam a capturar o comportamento pretendido do sistema. Esse comportamento pode ser expresso como funções, serviços ou tarefas, esboços dos fluxos de trabalho executados pelo sistema e outros requisitos comerciais ou de conformidade que o sistema deve atender ou necessário para executar. A Especificação de Requisitos Funcionais foi projetada para ser lida por um público em geral. Os leitores devem entender o sistema, mas nenhum conhecimento técnico específico deve ser necessário para entender o documento.

As questões trazidas pelos requisitos funcionais têm respostas que são independentes das implementações de uma solução para o problema do cliente. Descreve-se o que o sistema fará, sem discutir qual computador específico funcionará, qual linguagem de programação será implementado, estrutura de dados internos (PFLEEGER, 2004, p. 115).

Um requisito funcional define um sistema ou seus componentes. Ele descreve as funções ou os métodos que um software deve executar. Vamos utilizar o termo função para não confundir com os métodos de Classes. Embora quando forem feitas as codificações, veremos que estamos falando da mesma funcionalidade de programação. Uma função são as entradas (parâmetros de entrada), seu comportamento (processamento) e saídas, como cálculos, manipulação de dados, processos de negócios, interação do usuário ou qualquer outra funcionalidade. Os Requisitos Funcionais também são chamados de Especificação Funcional.

Segundo Fernandes (2017, p. 8), devemos eliminar dos requisitos funcionais quaisquer questões tecnológicas, pois, dessa forma, pode-se manter o espaço de soluções o mais amplo possível, aumentado as alternativas tecnológicas que poderão ser utilizadas durante o projeto.

Outro fato, explica Fernandes (2017, p. 68), é o requisito implícito que é incluído pela equipe de desenvolvimento, devido ao conhecimento do domínio, apesar de não ter sido explicitamente solicitado pelas partes interessadas.

Os requisitos funcionais de um sistema podem incluir, por exemplo, as seguintes características:

- Regras do negócio.
- Descrições de operações executadas por cada tela.
- Descrições de fluxos de trabalho executados pelo sistema.
- Correções, ajustes e cancelamentos de transações.
- Funções administrativas.
- Autenticação.
- Níveis de autorização.
- Acompanhamento de auditoria.
- Interfaces externas.
- Requisitos de certificação.
- Requisitos de relatório.
- Requisitos legais ou regulamentares.
- Descrições de dados a serem inseridos no sistema.
- Descrições de relatórios do sistema ou outras saídas.
- Quem pode inserir os dados no sistema.

O quadro a seguir apresente alguns exemplos de requisitos funcionais.

Quadro 1. Quadro de Requisitos Funcionais.

Requisitos Funcionais		
Código	**Nome**	**Descrição**
RF 1	Requisitos de interface.	
RF 1.1		O campo "valor do produto" aceita entrada de dados numéricos.
RF 1.2		O campo "data inicial" aceita apenas datas anteriores à data atual.
RF 1.3		A tela de consulta pode imprimir dados na tela na impressora.

Requisitos Funcionais		
Código	**Nome**	**Descrição**
RF 2	Requisitos de negócio.	
RF 2.1		Os dados devem ser inseridos antes que uma solicitação possa ser aprovada.
RF 2.2		Clicar no botão Aprovar move a solicitação para o fluxo de trabalho de aprovação.
RF 2.3		Todo o pessoal que utiliza o sistema será treinado de acordo com o SOP AA-101 interno.
RF 3	Requisitos de regulamentação/conformidade.	
RF 3.1		O banco de dados terá uma trilha de auditoria funcional.
RF 3.2		O sistema limitará o acesso a usuários autorizados.
RF 3.3		A planilha pode proteger dados com assinaturas eletrônicas.
RF 4	Requisitos de segurança.	
RF 4.1		Os membros do grupo Entrada de Dados podem inserir solicitações, mas não podem aprovar ou excluir solicitações.
RF 4.2		Os membros do grupo Gerentes podem inserir ou aprovar uma solicitação, mas não podem excluir solicitações.
RF 4.3		Os membros do grupo Administradores não podem inserir ou aprovar solicitações, mas podem excluir solicitações.

Fonte: autor.

Sommerville (2011, p. 60) menciona que, quando categorizados como requisitos de usuário, para serem compreendidos pelos usuários do sistema, os requisitos funcionais são normalmente definidos como abstratos. No entanto, requisitos de sistema funcionais mais específicos descrevem em detalhes as funções do sistema, suas entradas e saídas, exceções etc.

A seguir, serão apresentadas as diferenças entre os requisitos funcionais e os não funcionais:

Quadro 2. Diferenças entre Requisitos Funcionais e Não funcionais.

Parâmetros	Requisito funcional	Requisito não funcional
Pergunta: o que é isso?	Verbo	Atributos
É obrigatório.	Sim	Não
Como foi selecionado?	No caso de uso.	Um atributo de qualidade.
Resultado final.	Característica do produto	Propriedades do produto
Fácil de selecionar?	Sim	Não
Objetivo.	Verifica a funcionalidade do software.	Verifica o desempenho do software.
Área de foco dos usuários.	Requisitos.	Expectativas.
Documentação.	Descreva o que o produto faz.	Descreve como o produto funciona.
Tipo de teste.	Teste de integração, teste de API etc.	Testes de desempenho, estresse, usabilidade, testes de segurança etc.
Relacionado ao Produto	Características do produto	Propriedades do produto

Fonte: autor.

CAPÍTULO 3
Requisitos não funcionais

Segundo Sommerville (2011, p. 60), os requisitos não funcionais, como o nome sugere, são requisitos que não estão diretamente relacionados com as funcionalidades específicas oferecidos pelo sistema a seus usuários. Eles podem estar relacionados às propriedades emergentes do sistema, como confiabilidade, tempo de resposta e ocupação de área. Requisitos não funcionais podem afetar a arquitetura geral de um sistema em vez de apenas componentes individuais. Um único requisito não funcional, pode gerar uma série de requisitos funcionais relacionados que definam os serviços necessários no novo sistema. Por exemplo, tal como um requisito de proteção.

Um requisito não funcional é fundamental para garantir a operacionalidade e a eficácia de todo o sistema de software. Não disponibilizar esforços suficientemente necessários aos requisitos não funcionais pode gerar um sistema que vai falhar em satisfazer o objetivo do escopo, desagradando, assim, o cliente.

Outro cuidado em relação aos Requisitos Não funcionais está na possibilidade de o usuário encontrar maneiras de burlar ou contornar funcionalidades do sistema que não atenda suas necessidades.

As características dos Requisito Não Funcionais são mais críticas do que as dos Requisitos Funcionais. Deixar de atender prontamente um Requisito Não Funcional poderá levar a inutilização completa do Sistema.

Exemplos de requisitos não funcionais:

- Os usuários devem alterar a senha imediatamente após o primeiro login bem-sucedido.
- Somente funcionários autorizados poderão alterar dados dos demais funcionários.
- Toda tentativa malsucedida de um usuário de acessar a um item de dados deve ser registrada em uma trilha de auditoria.

» Um site deve ser capaz o suficiente para lidar com 20 milhões de transações por minuto sem afetar seu desempenho.

» Deverá haver portabilidade do software, pois, a mudança de sistema operacional não causará problema em seu funcionamento.

» A privacidade das informações, a exportação de tecnologias restritas, os direitos de propriedade intelectual etc. devem ser auditados.

» Toda tentativa de alteração não autorizada deve ser relatada ao administrador de segurança.

As principais vantagens dos Requisitos Não Funcionais são:

» Os requisitos não funcionais garantem que o sistema de software siga as regras legais e de conformidade.

» Eles garantem a confiabilidade, a disponibilidade e o desempenho do sistema de software

» Eles garantem uma boa experiência do usuário e facilidade de operação do software.

» Eles ajudam na formulação de políticas de segurança do sistema de software.

Aparência

Quando estamos desenvolvendo software sob medida, é comum nos atentar a identidade visual da empresa, como cores, logotipos, fontes etc. Fernandes (2017, p. 72) citou alguns exemplos de requisitos de aparência:

» O produto deve ter estilo igual ao dos outros produtos da empresa.

» O produto deve ser atrativo para usuários adolescentes.

» O produto deve ser identificável com a empresa onde vai ser usado.

Figura 2. Os automóveis são diferentes em função de requisitos funcionais e não funcionais aos quais atendem.

Fonte: Fernandes (2017, p. 73).

Usabilidade

A usabilidade está relacionada à facilidade de utilização, à curva de aprendizado, a fácil compreensão e a acessibilidade.

Segundo Fernandes (2017, p. 74), a facilidade de utilização está relacionada a eficiência de utilização do sistema, às interações com diversas mensagens emitidas aos usuários, aos mecanismos que evitam erros e conduzem ao "caminho feliz". A personalização está relacionada a capacidade de se adaptar o sistema aos gostos e necessidade dos usuários, como escolha de idioma, cores, fundos, ícones. A facilidade de aprendizagem está relacionada com a compreensibilidade que determina se os usuários percebem de forma intuitiva as funcionalidades do sistema e como operá-lo. Os requisitos de acessibilidade estão ligados a facilidade de o sistema oferecer aos usuários com algum tipo de deficiência, como visuais, físicas, auditivas, a operação normalmente de todas as suas funcionalidades.

Desempenho

Refere-se a capacidade de resposta a estímulos, como, consultas, gravações, ou qualquer outro evento. Ele está relacionado ao tempo de processamento de tarefas, à precisão de resultados, confiabilidade, disponibilidade, tolerância a falhas etc. Segundo Fernandes (2017, p. 75), o comportamento de um sistema de tempo real deve respeitar, além da funcionalidade pretendida, um conjunto de requisitos temporais, como por exemplo, o tempo limite para acionar um alarme no caso de uma situação de perigo deve ser cumprido, caso contrário, o resultado pode ser fatal à pessoas e patrimônios.

A precisão dos resultados refere-se aos cálculos efetuados pelo sistema, que são relacionados a hora, data, valores monetários, posicionamento geográfico, números precisos etc.

A tolerância a falha

É uma característica fundamental no que diz respeito à capacidade de o sistema manter o mesmo nível de funcionamento aceitável diante de situações complicadas, como a falha de alguns de seus componentes, pequenas quedas de rede etc.

A escalabilidade

É a capacidade de o sistema não falhar mesmo quando for demandado mais recursos, como acesso de usuários simultâneos, por exemplo. Outro fator de escalabilidade normalmente está ligado aos equipamentos. Normalmente, quando estamos desenvolvendo um sistema, certamente teremos que informar uma previsão de crescimento dos sistemas, dos dados, do processador, da memória etc. Entretanto, com o advento da computação em nuvem, esse fator está ficando mais fácil de contornar. Isso acontece, pois, empresas de datacenters oferecem facilidades para escalonar o hardware mesmo com o sistema já implantado.

Operacional

Está relacionado ao contexto que o sistema vai operar, como arquitetura (web, sistemas operacionais, dispositivos), ambientes físicos (ruídos, poeira, temperatura, luminosidade), mobilidade (ausência de sinais de comunicação, operar com uma das mãos etc.).

Fernandes (2017, p. 76) cita alguns exemplos:

- O produto deve operar embaixo de água até a profundidade de 30 metros.
- O produto deve carregar os dados em lote por meio de arquivo de texto.
- O produto deve exportar o *curriculum vitae* no formato *Europass*.

Manutenção e suporte

A manutenção do sistema está diretamente ligada à eficiência e padronização da codificação. Desenvolver sistemas que sejam fáceis de dar manutenção significa que, além de possuir códigos bem documentados, boa semântica, boas práticas etc., deva ser escrito de forma com que todos entendam. Veja o exemplo a seguir, o qual existem dois blocos de códigos que fazem o mesmo processamento, porém, por programadores diferentes:

Figura 3. Diferenças na escrita de códigos.

```
a = 2000;
b = 150;
c = 300;
d = 500
s = a + b + c - d

if(s > 5000)
     Calcula;
else
     Imprime;
```

```
salario_liquido = 2000;
hora_extra = 150;
beneficios = 300;
descontos = 500;
salario_bruto = salario_liquido +
                hora_extra +
                benefícios -
                descontos;

if(s > 5000)
     CalculaImposto;
else
     ImprimeSemImposto;
```

Fonte: Elaboração própria do autor.

Apesar de não ter usado comentários, o código dois ficou mais legível e fácil de fazer qualquer alteração.

Podemos classificar a manutenção em preventiva, corretiva, perfectiva e adaptativa.

Segurança

A segurança de um sistema está baseada nos pilares da segurança da informação: confidencialidade, disponibilidade e integridade. A disponibilidade pressupõe que o sistema estará com todas as suas funcionalidades sempre que forem requisitadas. Com a integridade, espera-se que as consistências das informações sejam as mesmas durante toda a existência da base de dados. A confidencialidade está ligada ao nível de acesso que cada usuário tem para acessar o sistema.

CAPÍTULO 4
Requisitos de usuários e de sistema

Não fazer separação entre os diferentes níveis de descrição gerará muitas dificuldades durante a fase de requisitos. Sommerville (2011, p. 57) faz uma distinção entre eles usando o termo “requisitos de usuário” e “requisitos de sistema”.

Requisitos de usuário e requisitos de sistema podem ser definidos como segue:

» Requisitos de usuário são declarações, para expressar os requisitos abstratos de alto nível, em uma linguagem natural, ou com diagramas, de quais serviços o sistema deverá fornecer a seus usuários e as restrições com as quais este deve operar. Segundo Fernandes (2017, p. 80), um requisito de usuário representa uma funcionalidade que se espera que o sistema forneça aos usuários ou uma restrição que se aplica a operações desse sistema.

» Os requisitos de usuários estão ligados ao domínio do problema e são representados sem muita formalidade, tanto na linguagem quanto em diagramas, facilitando a leitura e interpretação para as partes envolvidas e interessadas. Segundo Fernandes (2017, p. 80), uma designação mais adequada para requisitos de usuário seria “requisitos para as partes interessadas”, porém, requisitos de usuários é de uso muito generalizado.

» Os requisitos de sistema deverão ter a descrição de suas funcionalidades, como serviços e restrições de operacionalidade, expressando a descrição minuciosa que o sistema deverá fazer. Quem deve refletir exatamente o que o sistema deverá fazer será os documentos de requisitos, podendo fazer parte do contrato entre e o comprador e a empresa desenvolvedora de software.

Quadro 3. Separação dos requisitos por papéis.

Requisitos de usuários	Requisitos de Sistema
Gerentes.	Analistas do Sistema.
Usuários finais.	Desenvolvedores.
Fornecedores.	Arquiteto de Software.

Fonte: autor.

Segundo Fernandes (2017, p. 67), além de constituir especificações mais detalhadas, os requisitos de sistema são orientados ao domínio da solução e fornece informações aos engenheiros que irão ajudar a conceber o sistema, estando, portanto, em linguagem mais técnica do que os requisitos de usuários.

Exemplo de definição de requisitos de usuário:

» O sistema deve gerar relatórios gerenciais mensais que mostrem o custo dos medicamentos prescritos por cada clínica durante aquele mês. (SOMMERVILE 2011, p. 58).

» O usuário manipula arquivos criados por outros usuários. (FERNANDES, 2017, p. 67)

Especificação de requisitos de sistema:

» No último dia útil de cada mês, deve ser gerado um resumo dos medicamentos prescritos, seus custos e prescrições de cada clínica.

» Após as 17:30 do último dia útil do mês, o sistema deve gerar automaticamente o relatório para impressão.

» Um relatório será criado para cada clínica, listando os nomes dos medicamentos, o número total de prescrições, o número de doses prescritas e o custo total dos medicamentos prescritos.

» Se os medicamentos estão disponíveis em diferentes unidades de dosagem, devem ser gerados relatórios separados para cada unidade.

» O acesso aos relatórios de custos deve ser restrito a usuários autorizados em uma lista de controle de gerenciamento de acesso (SOMMERVILE 2011, p. 58).

» Os tipos de arquivo e os respectivos ícones são definidos pelo usuário.

» Cada tipo de arquivo está associado a um programa que processa e manipula os correspondentes arquivos.

» Quando um usuário clica em um ícone de um arquivo, esse arquivo é automaticamente aberto pelo programa que está associado a ele.

Na definição de requisitos de usuário e requisitos de sistema, devemos encontrar e classificar o que um “Requisito funcional e não funcional de usuário” e “Requisito funcional e não funcional de sistema”.

Os requisitos funcionais de um sistema variam de requisitos gerais (o que o sistema deve fazer), até requisitos muito específicos (refletem os sistemas e as formas de trabalho em uma organização).

Por exemplo, aqui estão os exemplos de requisitos funcionais para um sistema, usados para manter informações sobre os pacientes em tratamento por problemas de saúde mental, usado por Sommerville (2011, p. 59):

- Um usuário deve ser capaz de pesquisar as listas de agendamentos para todas as clínicas.
- O sistema deve gerar a cada dia, para cada clínica, a lista dos pacientes para as consultas daquele dia.
- Cada membro da equipe que usa o sistema deve ser identificado apenas por seu número de oito dígitos.

Esses requisitos funcionais dos usuários definem os recursos específicos a serem fornecidos pelo sistema. Eles foram retirados do documento de requisitos de usuário e mostram que os requisitos funcionais podem ser escritos em diferentes níveis de detalhamento (contrastar requisitos 1 e 3).

A imprecisão na especificação de requisitos é a causa de muitos problemas da engenharia de software. É compreensível que um desenvolvedor de sistemas interprete um requisito ambíguo de uma maneira que simplifique sua implementação. Muitas vezes, porém, essa não é a preferência do cliente, sendo necessário, então, estabelecer novos requisitos e fazer alterações no sistema. Naturalmente, esse procedimento gera atrasos de entrega e aumenta os custos.

Por exemplo, o primeiro requisito hipotético, para o sistema sugerido por Sommerville (2011, p. 60), é de um sistema usado por pacientes com problema de saúde mental, no qual um usuário deve ser capaz de buscar as listas de agendamentos para todas as clínicas.

Como as clínicas estão interligadas, eles poderão marcar uma consulta sem se deslocar para outras, ou seja, quando alguém fizer uma consulta, ela será disparada paras todas as clínicas. Porém, isso não ficou explícito nas especificações do requisito, podendo causar má interpretação por parte dos

desenvolvedores. Nesse caso especificamente, os desenvolvedores poderão implementar uma pesquisa em que o usuário tenha de escolher uma clínica e, em seguida, realizar a pesquisa. Isso fará com que o usuário tenha que realizar muitas operações antes de chegar ao resultado esperado.

ENGENHARIA DE REQUISITOS

UNIDADE II

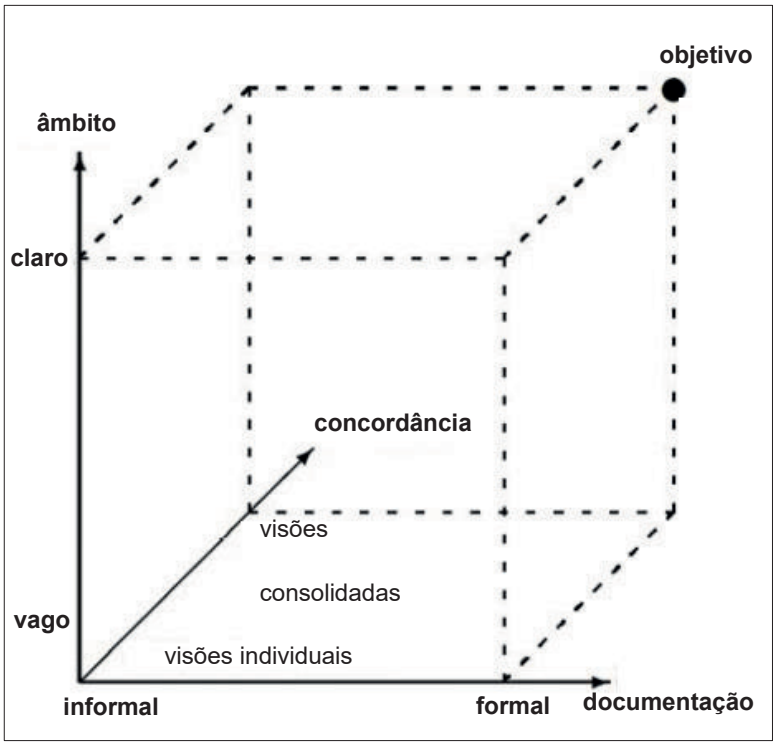
CAPÍTULO 1

Especificação de requisitos

Processos de engenharia de requisitos podem incluir quatro atividades de alto nível. Elas visam avaliar se o sistema é útil para a empresa (**estudo de viabilidade**), descobrindo requisitos (**elicitação e análise**), convertendo-os em alguma forma-padrão (**especificação**), e verificar se os requisitos realmente definem o sistema que o cliente quer (**validação**) (SOMMERILLE 2011, p. 69).

Fernandes (2017, p. 92) ilustra muito bem as dimensões da Engenharia de Requisitos:

Figura 4. As três dimensões da engenharia de requisitos.



Fonte: Fernandes (2017, p. 92).

Sommerville (2011, p. 65) refere-se como sendo o processo que escrever os requisitos, tanto de usuários, quanto de sistemas, em documentos de forma clara e objetivas, inequívocas, de fácil compreensão, completos e consistentes.

Existem várias formas e técnicas para descrever os requisitos, dentre elas:

Quadro 4. Formas de escrever uma especificação de requisitos de sistema.

Notação	Descrição
Sentenças em linguagem natural.	Os requisitos são escritos em frases numeradas em linguagem natural e cada frase deve expressar um requisito.
Linguagem natural estruturada.	Os requisitos são escritos em linguagem natural em um formulário padrão ou *template*.
Linguagem de descrição de projeto.	Essa abordagem usa uma linguagem como de programação, mas com características mais abstratas, para especificar os requisitos, definindo um modelo operacional do sistema.
Notações gráficas.	Para definição dos requisitos funcionais para o sistema, são usados modelos gráficos, suplementados por anotações de texto; diagramas de caso de uso e de sequência da UML são comumente usados.
Especificações matemáticas	Essas notações são baseadas em conceitos matemáticos, como máquinas de estado finito ou conjuntos.

Fonte: Sommerville (2011, p. 67).

Especificação em linguagem natural

A linguagem natural é sem dúvida a forma mais utilizada para escrever os requisitos de software. Sua vantagem contempla ser intuitiva, universal e expressiva. Porém, quando se trata de projetos maiores em que, normalmente, envolve uma equipe mais diversificada e seu entendimento precisa de precisão, ela se torna ambígua, dependente do entendimento do leitor, potencialmente vaga, portanto, inviável. Precisamos que os requisitos sejam especificados com exatidão, porém, temos que contar que a linguagem natural sempre será a mais utilizada. Para isso, podemos minimizar seus impactos, conforme algumas técnicas sugeridas por Sommerville (2011, p. 67):

» É válido a criação de um novo padrão de escrita qualquer desde que seu formato garanta as definições necessários dos requisitos e sua aderência. Entretanto, é altamente recomendável que seu significado seja expresso em uma única frase, evitando mal-entendidos, ambiguidades etc.

» Deve ser muito claro quando for definir requisitos obrigatórios e desejáveis. No entendimento do usuário, os requisitos obrigatórios são aqueles de “devem” e os requisitos desejáveis são aqueles que “podem”. Os exemplos mais marcantes são os campos em sistemas de gestão. Em um módulo do sistema que emite Nota Fiscal, o

número da Nota deverá ser necessário o preenchimento. Já o campo Observação, poderá ser preenchido caso seja necessário.

» Destaque as partes fundamentais do requisito.

» Jamais assuma que os leitores dos requisitos irão compreender as linguagens técnicas. Segundo Sommerville (2011, p. 67), palavras como “módulo” e “arquitetura” são mal interpretadas ou de dubio sentido. Evite, portanto, o uso de jargões, siglas e acrônimos.

» É importante, sempre que possível, associar uma lógica para cada requisito de usuário, pois, deve explicar por que o requisito foi incluído, e é particularmente útil quando os requisitos são alterados, uma vez que pode ajudar a decidir sobre quais mudanças seriam indesejáveis (SOMMERVILLE, p. 67).

Domínio básico da escrita

Fernandes (2017, p. 176) exalta a importância de se dominar a escrita como ortografia, pontuação, concordâncias etc., para que um nível bem aperfeiçoado seja atingido, é importante praticar muito. Essa prática garante habilidades em manter o entendimento organizado, sempre que houver mudanças. Outro ponto a considerar são que as linguagens técnicas devem ser claras, objetivas e simples, evitando metáforas, e deve ser redigida em estilo impessoal.

Formatos padronizados

Por existirem vários formatos, podemos seguir algumas frases que representam o contexto que se está abstraindo. Fernandes (2017, p. 177) sugere alguns exemplos para os requisitos quando o sujeito for o usuário, as funcionalidades, os testes, e o conceito:

» Usuário: o recepcionista do hotel.

» Funcionalidade: “deve visualizar”.

» Conceito: o número do quarto do hóspede.

» Teste: dois segundos após efetuar o pedido.

Quando o sujeito for o sistema em desenvolvimento ou entidade de concepção, segundo Fernandes (2017, p. 178), no resultado a ser atingido, o verbo será as funcionalidades e o objeto as descrições, por exemplo:

- Sistema ou entidade: sinal da bateria.
- Funcionalidades: deve ligar.
- Descrição: quando a carga for inferior a 20 *mAh*.

Especificações estruturadas

Uma outra forma de padronizar a expressividade da linguagem natural, na escrita de requisitos de sistema, é a linguagem natural estruturada, na qual todos os requisitos obedecem a uma forma padrão, por meio de *templates* ou formulários estruturados.

Por possuírem, normalmente, padrões de escrita e destaques de palavras reservadas, indentações, destaques muitas construções são feitas no formato de linguagem de programação.

Segundo Peters (2010, p. 105), as especificações de requisitos devem identificar o objetivo, o escopo, os acrônimos, as abreviações, as referências, e a visão geral do documento de requisitos.

Os objetivos irão especificar as intenções do cliente que está contratando o projeto, ou seja, o produto de software. Os acrônimos e símbolos devem ser explicados e detalhados de forma a todos entenderem, sem ambiguidades ou outras anomalias. Uma das principais razões pelas quais os projetos de software falham é porque os requisitos do projeto não foram capturados corretamente. Geralmente, durante o ciclo de vida do projeto, as demandas continuam variando e isso também pode ter um impacto na obtenção de requisitos adequados.

O resultado dos processos da Especificação de Requisitos é o Documento de Requisitos.

A especificação de requisitos abrange essas tarefas para determinar as necessidades de uma solução ou produto de software proposto, geralmente, envolvendo requisitos de várias partes interessadas associadas à solução. A especificação de requisitos é um componente essencial no ciclo de vida de desenvolvimento de software e geralmente é a etapa inicial antes do início

do projeto. A especificação de requisitos é muito cara, exigindo enormes investimentos em recursos escassos, sistemas e processos associados.

Este artigo se concentra em uma metodologia adotada durante a fase de requisitos e especificação funcional de um projeto. O modelo escolhido para a engenharia de requisitos foi fundamentado em uma combinação de técnicas seis sigma e um conjunto de melhores práticas adotadas dentro da organização.

Acesse: https://www.stickyminds.com/article/requirements-engineering-our-best-practices.

CAPÍTULO 2
Documento de requisitos

Em gerenciamento de projetos, seja ele de software ou não, todo escopo está diretamente baseado nos requisitos levantados pelos envolvidos. E, para que tudo seja formalmente documentado, várias etapas do projeto possuem sua documentação específica. Em projetos software não poderia ser diferente.

Segundo Sommerville (2011, p. 63), é uma declaração oficial de que os desenvolvedores do sistema devem implementar. Deve incluir tanto os requisitos de usuário para um sistema quanto uma especificação detalhada dos requisitos de sistema.

Peter (2010, p. 103) menciona que a especificação de requisitos (ERS) de software é a descrição de um produto de software, programa ou conjunto de programas específicos que executam uma série de funções no ambiente de destino. Ela emerge dos componentes de análise dos problemas e seus primeiros esboços são escritos durante o processo de decomposição de resultados.

Por muito tempo, as boas práticas de documentos de requisitos foram utilizadas para orientar os desenvolvedores. Entretanto, hoje os métodos ágeis estão tornando os formatos de documentos de requisitos antigos, obsoletos. Segundo Sommerville (2011, p. 63), arquitetos de metodologias ágeis argumentam que os requisitos mudam tão rapidamente que um documento de requisitos já está ultrapassado assim que termina de ser escrito. Portanto, o esforço é, em grande parte, desperdiçado. Em vez de um documento formal, abordagens como a *Extreme Programming* coletam os requisitos de usuário de forma incremental e escrevem-nos em cartões como estórias de usuário.

Essa abordagem do *Extreme Programming* pode colocar alguma agilidade onde os sistemas de negócios possuem seus requisitos instáveis durante o desenvolvimento inicial. No início do desenvolvimento, a demanda será maior e é normal que haja muita alteração de requisitos. Depois, com as entregas feitas, essas alterações diminuem e as atualizações dos requisitos poderão ser feitas com mais tranquilidade. Sommerville (2011, p. 63) acredita que ainda seja útil escrever um pequeno documento de apoio no qual estejam definidos os requisitos de negócio e de confiança para o sistema.

Segundo o PMBOK (2017, p. 116), a documentação dos requisitos descreve como os requisitos individuais atendem às necessidades do negócio para o projeto, eles podem começar em um alto nível e tornarem-se progressivamente mais detalhados conforme mais informações sobre estes são conhecidas.

Antes das linhas de base de um escopo serem estabelecidas, os requisitos devem ser não ambíguos, mensuráveis e passíveis de testes, rastreáveis, completos, consistentes e aceitáveis para as principais partes interessadas. O formato de um documento de requisitos pode variar de uma simples lista categorizada por partes interessadas e prioridades a formas mais elaboradas contendo um resumo executivo, descrições detalhadas e anexos.

O PMBOK (2017, p. 116) sugere que os componentes da documentação dos requisitos podem incluir, mas não estão limitados, a:

1. **Requisitos de negócios**:
   - Objetivos do negócio e do projeto para permitir rastreamento.
   - Regras de negócios para a organização executora.
   - Os princípios e diretrizes da organização.
2. **Requisitos das partes interessadas, incluindo**:
   - Impactos em outras áreas organizacionais.
   - Impactos em outras entidades internas ou externas à organização.
   - Requisitos de comunicação com as partes interessadas e de relatórios.
3. **Requisitos de solução, incluindo**:
   - Requisitos funcionais e não funcionais.
   - Requisitos tecnológicos e de conformidade com padrões.
   - Requisitos de suporte e treinamento.
   - Requisitos de qualidade.
   - Requisitos de relatos etc.
4. **Requisitos do projeto, tais como**:
   - Níveis de serviço, desempenho, segurança, conformidade etc.
   - Critérios de aceitação.

5. **Requisitos de transição**:

› Premissas, dependências e restrições dos requisitos.

O documento de requisitos percorre uma diversidade muito grande de usuários. Esse usuários possuem visões diferentes, departamentos diferentes, conhecimento e cargos diferentes. Eles estão dispostos em um projeto desde a alta administração da organização, que está pagando pelo sistema, até os engenheiros e arquitetos responsáveis pelo desenvolvimento do software. O quadro a seguir mostra os possíveis usuários do documento e como eles o utilizam.

Quadro 5. Usuários dos documentos de requisitos.

Papéis	Descrição
Clientes do sistema	Especificam e leem os requisitos para verificar se estes satisfazem suas necessidades. Os clientes especificam as alterações nos requisitos.
Gerentes	Usam o documento de requisitos para planejar uma proposta para o sistema e para planejar o processo de desenvolvimento do sistema.
Engenheiros do sistema	Usam os requisitos para entender o sistema que será desenvolvido.
Engenheiros de teste de sistema	Usam os requisitos para desenvolver testes de validação do sistema.
Engenheiros de manutenção de sistema	Usam os requisitos para entender o sistema e os relacionamentos entre suas partes.

Fonte: Sommerville (2011, p. 64).

A equipe do projeto de software será a responsável pela edição do documentos de requisitos. Essa tarefa é bastante minuciosa e requer um detalhamento ao nível extremo. Segundo Fernandes (2017, p. 185), definir uma estrutura genérica, revela-se importante graças as diversidades de sistemas e projetos que poderão ser encontrados. Existe uma liberdade excessiva no desenvolvimento de documentação de requisitos que poderá variar de caso para caso, devido ao grau de complexidade. Nessa fase, podemos agradar as pessoas que preferem seguir uma estrutura pré-definida e desagradar o pessoal das metodologias ágeis, como já mencionamos.

A estrutura a seguir foi adotada por Fernandes (2017, p. 185):

Quadro 6. Estrutura de documentação.

Instigadores do Projeto	
1	Propósito do Sistema
2	Patrocinadores e *stakeholders*
3	Usuários do Sistema
Restrições do Projeto	
4	Restrições obrigatórias
5	Taxionomia e definições
6	Fatos e assunções relevantes

Requisitos Funcionais	
7	Âmbito de trabalho
8	Âmbito do Sistema
9	Requisitos funcionais e de dados
Requisitos não funcionais	
10	Aparência
11	Usabilidade
12	Desempenho
13	Operacionais
14	Manutenção e suporte
15	Segurança
16	Culturais e políticos
17	Legais
Tópicos do projeto	
18	Tópicos abertos
19	Soluções
20	Novos Problemas
21	Tarefas
22	Migrações
23	Riscos

Fernandes (2017, p. 185).

» **Propósito do sistema**: descreve o propósito, o contexto do domínio do sistema, os objetivos e o motivo que levou o cliente a pedir o sistema.

» **Patrocinadores e *stakeholders***: deve ser informado, em primeiro lugar, o nome, o cargo e o tipo de influência dos patrocinadores, acionistas ou donos do sistema, pois, são as pessoas que investiram e esperam uma atenção diferenciada e um retorno garantido. Em seguida, os demais *stakeholders*.

» **Usuários do sistema**: é criada uma lista dos usuários, seus níveis de acesso, duração e período de acesso. Também é importante descrever se haverá usuários temporários ou visitantes.

» **Restrições obrigatórias**: segundo Fernandes (2017, p. 184), nesta seção devem ser inseridos as soluções ou tecnologias adotadas, os motivos que levaram a adotá-las, os sistemas e a forma com que os sistemas vão interagir, os ambientes em que os usuários operam, as bibliotecas de terceiros, restrições de prazos e preços, em que, em alguns caso, o cliente pode pedir mais restrições do que foram acordadas.

» **Taxionomia e definições**: nessa seção, devemos incluir um dicionário ou glossário com o significado de todas as nomenclaturas importantes para o entendimento do domínio do sistema.

» **Fatos e assunções relevantes**: aqui devemos todos os fatos relevantes que possa contribuir com a gestão do conhecimento, relacionado às experiências. Nesse momento, uma equipe madura e entrosada tem condições de assegurar, aos novos membros, um ritmo de trabalho desejável, logo que entrar na equipe.

» **Âmbito do sistema**: são incluídos descrições textuais e diagrama de uso de caso.

» **Requisitos funcionais e de dados**: segundo Fernandes (2011, p. 189), cada requisito deve ser especificado de acordo com os seguintes campos:

› Identificador único: atributo usado como referência unívoca (chave primária). Normalmente um contador numérico.

› Tipo: indicação numérica da seção do documento onde está incluído.

› Caso de uso: lista de casos de uso que necessitam do requisito.

› Descrição: frase que escreve a funcionalidade oferecida pelo requisito.

› Fonte: indicações da origem do requisito.

› Satisfação do cliente: nível de satisfação se o requisito for incluído no sistema.

› Prioridade: nível de prioridade do requisito.

› Conflitos: identificação de outros requisitos que estejam em conflito com este.

› Datas: registro das datas em que o requisito foi criado e alterado.

» **Aparência**: inclui requisitos não funcionais ligados a aparência do sistema, estilo etc.

» **Usabilidade**: são requisitos não funcionais ligados a facilidade de utilização, aprendizado etc.

» **Desempenho**: inclui requisitos funcionais ligados ao desempenho dos processamentos do sistema. Também inclui a confiabilidade, robustez, precisão etc.

» **Operacionais**: contexto ambiental ou tecnológico em que o sistema vai funcionar.

» **Manutenção e suporte**: relacionado ao suporte e manutenção do sistema.

» **Segurança**: seção inclui a descrição dos fatores ligados aos pilares da Segurança da Informação. Deve ser consultado a política de Segurança de Informação da empresa para confrontar com as funcionalidade do sistema.

» **Culturais e políticos, legais**: relacionados aos tópicos de requisitos não funcionais de cultura, política e legislação.

» **Tópicos abertos**: tópicos abordados na fase de análise e não foram resolvidos.

» **Soluções**: alguma solução já pronta, seja ela um sistema novo ou um código já pronto.

» **Novos problemas**: segundo Fernandes (20177, p. 191), nessa seção, deve se antecipar detecção de potenciais conflitos, descrever o modo que o sistema vai influenciar o ambiente de trabalho, como o sistema fará interfaces com outros sistemas.

» **Tarefas**: descrevem os processos de desenvolvimento estabelecido para uso do contexto do fabricante, quando tiver de ser modificado, em função de circunstâncias especiais associadas ao projeto, podendo fazer estimativas de tempo e recursos (FERNANDES, 2017, p. 192).

» **Migrações**: descreve, basicamente, informações técnicas e procedimentos aleatórios, como será desligado o sistema, ligado e iniciado o novo.

Quando existe uma diversidade de usuários, como por exemplo níveis de conhecimento, hierarquia, faixa etária etc., é um bom indicativo de que o documento de requisitos precisa variar na forma como será feita a comunicação para os clientes, desenvolvedores ou arquitetos. O detalhamento deve ser discutido e dependerá do sistema que está sendo desenvolvido.

Segundo Sommerville (2011, p. 64), os sistemas críticos precisam ter requisitos detalhados, porque a segurança e a proteção devem ser analisadas em detalhes. Quando o sistema está sendo desenvolvido por uma companhia separada (por exemplo, por meio de terceirização), as especificações de sistema devem ser detalhadas e precisas. Se um processo interno de desenvolvimento iterativo é usado, o documento de requisitos pode ser muito menos detalhado e quaisquer ambiguidades podem ser resolvidas durante o desenvolvimento do sistema.

O quadro a seguir mostra uma possível organização de um documento de requisitos baseada em uma norma IEEE para documentos de requisitos. Essa é uma norma genérica que pode ser adaptada para usos específicos.

Quadro 7. A estrutura de um documento de requisitos.

Capítulo	Descrição
Prefácio	Deve definir os possíveis leitores do documento e descrever seu histórico de versões, incluindo uma justificativa para a criação de uma nova versão e um resumo das mudanças feitas em cada versão.
Introdução	Necessidades para o sistema, as funções e integrações, como atingirá os objetivos de quem encomendou o sistema.
Glossário	Definir os termos técnicos usados no documento. Você não deve fazer suposições sobre a experiência ou o conhecimento do leitor.
Definição de requisitos de usuário	Deve descrever os serviços fornecidos ao usuário, os requisitos não funcionais, descrição pode usar a linguagem natural, diagramas ou outras notações compreensíveis para os clientes.
Arquitetura do sistema	Visão geral em alto nível da arquitetura do sistema previsto; componentes reutilizados devem ser destacados.
Especificação de requisitos do sistema	Detalhes dos requisitos funcionais, não funcionais e interfaces com outros sistemas.
Modelos do sistema	Modelos gráficos do sistema. Exemplos de possíveis modelos são modelos de objetos, modelos de fluxo de dados ou modelos semânticos de dados.
Evolução do sistema	Os pressupostos fundamentais em que o sistema se baseia, bem como quaisquer mudanças previstas, em decorrência da evolução de hardware, de mudanças nas necessidades do usuário etc.
Apêndices	Deve fornecer informações detalhadas e específicas relacionadas à aplicação em desenvolvimento, além de descrições de hardware e banco de dados, por exemplo.
Índice	Vários índices podem ser incluídos no documento. Pode haver, além de um índice alfabético normal, um índice de diagramas, de funções, entre outros pertinentes.

Fonte: Sommerville (2011, p. 5).

CAPÍTULO 3
Validação de requisitos

Segundo Sommerville (2011, p. 76), a validação de requisitos é o processo pelo qual se verifica se os requisitos definem o sistema que o cliente realmente quer, além de se sobrepor à análise, pois, está preocupada em encontrar problemas com os requisitos.

A validação dos requisitos é importante porque é nesse momento que erros são encontrados e podem ser evitados altos custos de retrabalho, evitando encontrá-los durante o desenvolvimento do sistema ou pior, quando o sistema já estiver em funcionamento. Os custos para reparar problemas de requisitos que acontecem por mudanças nos requisitos e, por sua vez, afetarão a implementação que tem seu impacto muito maior, além executar os ciclos de testes e liberações.

A validação de requisitos facilita na resolução dos conflitos entre os diferentes interessados do sistema, devido à incompletude ou qualquer incompatibilidade entre os requisitos dentro dos constrangimentos financeiros disponíveis.

Outra explicação é de Pfleeger (2004, p. 142), que menciona a validação de requisitos como um processo que determina que a especificação é consistente com a definição dos requisitos, ou seja, a validação assegura que os requisitas atenderão as necessidades dos clientes.

O processo de validação deverá passar por formas diversificadas de validações, sendo elas sugeridas por Sommerville (2011, p. 77):

- **Verificações de consistência**: na documentação, os requisitos não devem entrar em conflitos e nem descrições diferentes para a mesma função do sistema.
- **Verificações de completude**: todas as funções e restrições pretendidas pelo usuário devem estar nos documentos de requisitos.
- **Verificações de realismo**: os requisitos devem ser verificados sempre para assegurar sua implementação. Para isso, devem ser utilizadas tecnologias e conhecimentos existentes, sem desconsiderar o orçamento e o cronograma do desenvolvimento do sistema.

Verificabilidade

Um conjunto de testes devem ser demonstrados, sob a documentação dos requisitos, a fim de evitar possíveis conflitos entre o cliente e o contratante.

Além disso, outras técnicas poderão ser utilizadas para validação dos requisitos:

» **Revisões de requisitos**: uma equipe de revisores deverá analisar sistematicamente possíveis erros e inconsistências.

Prototipação

Segundo Sommerville (2011, p. 77), nessa abordagem para validação, um modelo executável do sistema em questão é gerado e demonstrado para os usuários finais e clientes. Estes podem experimentar o modelo para verificar se ele atende a suas reais necessidades. Requisitos prototipagem é uma técnica fundamental para a validação de requisitos dos usuários, uma vez que representa o funcionamento de um sistema real a ser construído e auxilia na identificação dos requisitos que não foram devidamente compreendidos.

Protótipos facilitam nos requisitos de validação, fornecendo informações valiosas sobre o sistema. Protótipos são uma boa ferramenta para os requisitos de validação, especialmente quando você não está bastante confiante de que você tem um bom conjunto de requisitos. Os dois tipos de protótipos mais utilizadas são:

» **Protótipos descartáveis**: protótipos descartáveis são eliminados após o *feedback* do usuário, uma vez que os requisitos iniciais estabelecidos são construídos no protótipo. Ele ajuda na resolução de requisitos de conflitos entre a equipe de desenvolvimento e os clientes, tendo *feedback* sobre o protótipo e, se ambos acordarem, um conjunto definido de requisitos, então, o protótipo é descartado e os requisitos são oficializados na documentação.

» **Protótipos evolutivos:** são construídos a partir requisitos iniciais e gradualmente refinamentos são feitos, dependendo de feedback do usuário. “Prototipagem evolucionária” é realizada em um conjunto de requisitos resolvidos e está sujeito a restrições de qualidade imposta no desenvolvimento de software.

Geração de casos de teste

Segundo Sommerville (2011, p. 77), os requisitos devem ser testáveis pois, se os testes forem concebidos como parte do processo de validação, problemas de requisitos serão revelados. O grau de dificuldade para a projeção de um teste de requisitos será proporcional à sua implementação, ou seja, se for difícil ou impossível de efetuar testes, sua implementação deverá ser considerada. O *Extreme Programming* contempla o desenvolvimento de testes antes de qualquer codificação.

O objetivo de executar o teste é requisito para assegurar a validação dos requisitos em vez de validá-los em sistema software. Para isso, casos de teste são gerados para todos os requisitos estabelecidos, o tempo para escrever manuais de ensaio e/ou recursos econômicos.

Os custos do processo de Validação de Requisitos deverão estar incluídos no custo de validação de requisitos. Requisitos auxiliares de ensaio na identificação de requisitos ambíguos ou incompletos de um modo que, se houver algum problema, ocorre durante a execução de um processo de teste para qualquer exigência específica, que dá uma indicação de que há algum problema com esta exigência.

Para Pfleeger (2004, p. 142), a validação se constitui em assegurar a rastreabilidade entre dois documentos de requisitos. Primeiro, é assegurado que cada especificação pode ser rastreada para um requisito no documento de definições. Em seguida, é verificada a definição, a fim de ver se cada requisito pode ser rastreado para especificação. Algumas delas são:

Quadro 8. Técnicas de validação de requisitos.

Técnicas manuais	**Leitura**
	Referência cruzada manual.
	Entrevistas.
	Revisões.
	Listas de verificações.
	Modelos manuais para verificar funções e relações.
	Cenários.
	Provas matemáticas.
Técnicas automatizadas	Referência cruzada matemática.
	Modelos automatizados para ativar funções.
	Protótipos.

Fonte: Fernandes (2019, p. 143).

Vale lembrar que a validação vai além de rastreabilidade. Para garantir que o software efetuará suas funcionalidades conforme foi acordado com clientes e usuários, é importante também garantir a forma de aceite com que os usuários e os clientes ficarão satisfeitos. Senão, o problema será passado para o desenvolvimento e implementações.

Segundo Pfleeger (2004, p. 142), uma maneira simples de conferir os requisitos é revisá-los com representantes da equipe de gestão do projeto e a equipe do cliente. Na lista, os representantes devem incluir quem vai operar o sistema, quem vai preparar as entradas e as saídas. Essa revisão envolve:

- Revisar metas e objetivos estabelecidos para o sistema.
- Comparar os requisitos com as metas e objetivos.
- Descrever o ambiente que o sistema irá operar.
- Avaliar e documentar riscos, se houver a probabilidade.
- Conversar sobre os testes do sistema.

Completando a fase de revisão, clientes, usuários, projetistas já poderão ficar tranquilos em relação às especificações de requisitos, tendo o cliente em mãos um documento formalizando exatamente o que o sistema fará.

CAPÍTULO 4
Gerenciamento de requisitos

Quando se trata de requisitos de software, os envolvidos no projeto têm que ter em mente que as mudanças serão inevitáveis. Ao longo do tempo, inúmeras tentativas de tornar o ciclo de desenvolvimento de software mais eficientes foram testados. Alguns foram aprovados e seguiram com suas adaptações e outros, simplesmente desapareceram.

Segundo Sommerville (2011, p. 77), os requisitos para sistemas de software de grande porte estão em constante alterações e uma das razões para isso é que esses sistemas são desenvolvidos para enfrentar problemas que não podem ser completamente definidos e, por isso, os requisitos de software são obrigados a ser incompletos.

Após a implantação, sempre surgirão novos requisitos. Isso é normal, pois, novas ideias de negócios poderão surgir com a plena utilização do sistema, uma vez que os usuários e clientes ganharão experiências e se sentirão mais à vontade diante do reflexo, do seu processo de negócio, no sistema.

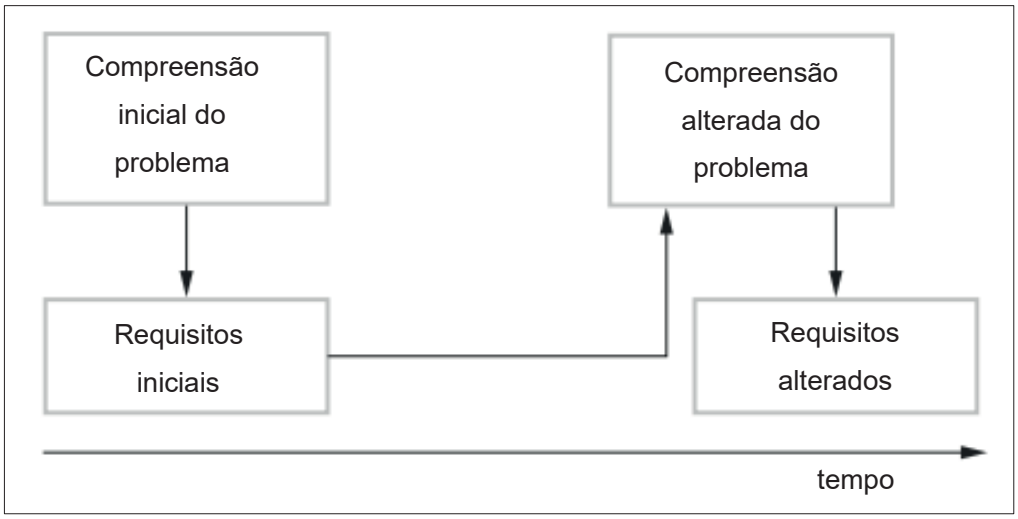
Sommerville (2011, p. 77) cita algumas razões de haver mudanças nos requisitos após a implantação do sistema:

- Alterações no ambiente técnico e de negócios, a instalação de um novo hardware, necessidade de uma nova API para integrar com novos sistemas, alterações no processo de negócios para tornar a empresa mais competitiva, adequações à legislação etc.
- Algumas restrições orçamentárias e organizacionais podem entrar em conflito com os requisitos de usuários.
- A diversidade de usuários em sistemas de grande porte poderá aumentar a necessidade com as diferenças de requisitos.

Para que o projeto de software não perca o controle das constantes modificações nos requisitos, é muito importante que seu gerenciamento seja feito de forma eficiente, ágil e confiável, e que a equipe envolvida sempre esteja disposta a aceitar as mudanças que agregarão novos recursos ao sistema.

Segundo Sommerville (2011, p. 78), gerenciar os requisitos é compreender e controlar as mudanças nos requisitos do sistema. Para isso, é muito importante estar caminhando juntos com as necessidade individuais, manter as ligações entre as necessidade dependentes, para mensurar o impacto dessas mudanças e acompanhar a evolução de novas percepções dos problemas.

Figura 5. Evolução dos requisitos.



Fonte: Sommerville (2011, p. 78).

Para isso, é necessário estabelecer um processo formal de gerenciamento, que deverá começar tão logo uma versão preliminar dos documentos de requisitos estiver disponível, já na elicitação dos requisitos. Muitas equipes utilizam ferramentas de controle de versão se o projeto de gerenciamento de requisitos for feito em documentos. Também existem ferramentas próprias para esse propósito.

Todo gerenciamento de projeto começa pelo planejamento e com o gerenciamento de requisitos não seria diferente. O Planejamento de Requisitos é a primeira fase no processo de gerenciamento de requisitos. Por meio dele, será possível definir o nível de detalhamento que deverá ser aplicado no gerenciamento. Sabemos que, quanto mais detalhado os requisitos de um produto ou projeto, menos chances de o projeto falhar teremos. Por isso a importância dessa etapa decidirá sobre alguns pontos importantes:

» **Identificação de requisitos**: cada requisito deverá ter uma identificação única para que possa ser rastreado e "*versionado*[2]".

» **Processo de gerenciamento de mudanças**: esse processo avaliará o impacto e o custo das mudanças.

» **Políticas de rastreabilidade**: estabelece como e quais devem ser os relacionamentos entre os requisitos e o projeto de sistema. Definem os relacionamentos entre cada requisito e entre os requisitos e o projeto de sistema que deve ser registrado.

» **Ferramenta de apoio**: como gerenciar os requisitos é um processo que envolverá grandes quantidades de informação, ferramentas auxiliares deverão ser decididas nesse momento. Elas vão desde software proprietário, banco de dados e planilha.

Como mencionado anteriormente, o pessoal com profundo envolvimento com metodologias ágeis, como o *Extreme Programming*, tem uma certa aversão aos métodos tradicionais de gerenciamento de requisitos. Por isso, automatizar o processo de gerenciamento será fundamental.

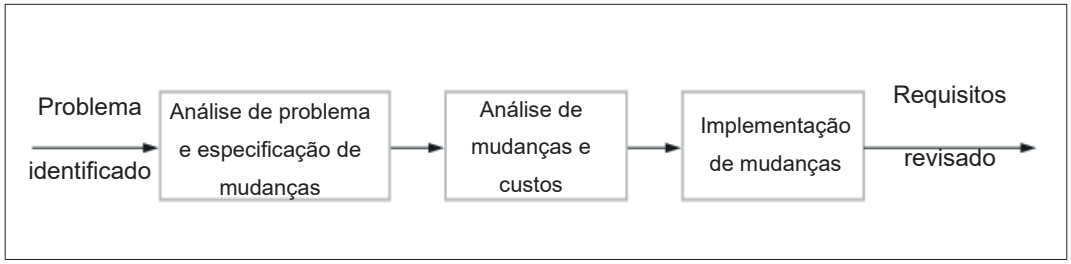
Para que os métodos de gerenciamento ganhem agilidade, é importante gerenciar as mudanças com ferramentas ativas, gerenciar a rastreabilidade para descobrir os requisitos relacionados e suas modificações e, ao final, manter todas essas informações armazenadas em banco de dados.

Vale lembrar que sistemas de pequeno porte têm seu orçamento reduzido, e gerenciar todo o processo de gerenciamento de requisitos com ferramentas caras pode tornar o projeto inviável. Entretanto, mesmo para esses sistemas, é muito importante que haja o gerenciamento por meio de planilhas, banco de dados de pacotes de escritório, processadores de textos.

Logo que o documento de requisitos seja aprovado, o gerenciamento de mudanças deve estar pronto para entrar em ação. Segundo Sommerville (2011, p. 79), nessa fase, deve decidir se haverá benefícios das implementações das novas implementações e também se os custos se justificarão. Ter um processo formal de gerenciamento de mudanças nos traz muitas vantagens, pois toda as mudanças serão tratadas de forma controladas e consistentes.

Sommerville (2011, p. 79) propõe três estágios principais no processo de gerenciamento de mudanças:

Figura 6. Gerenciamento de mudança de requisitos.

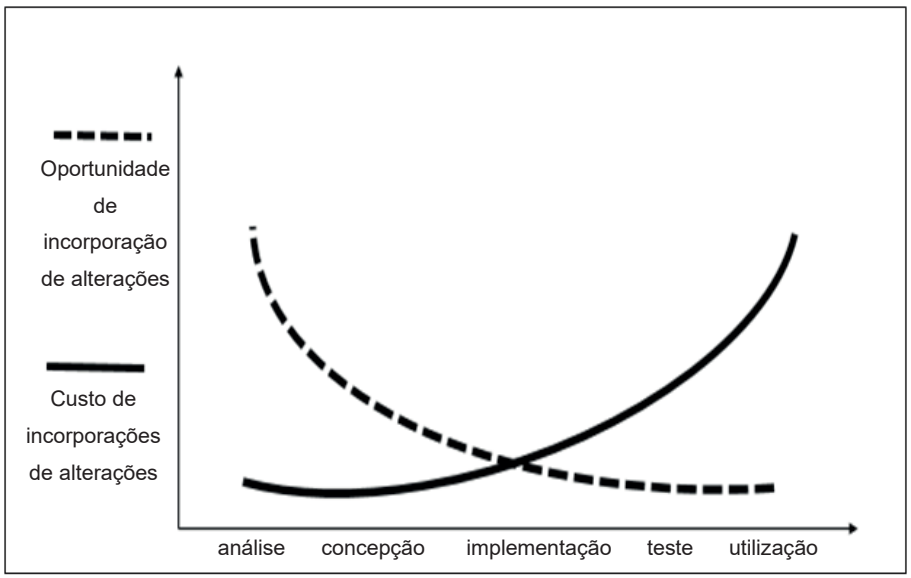


Fonte: Sommerville (2011, p. 79).

A análise de problema e especificação de mudanças inicia com um requisito identificado ou uma intensão de mudança. Para seguir adiante, analisa-se o problema para garantira sua viabilidade e, em seguida, decide se a proposta retorna para quem enviou ou entra para a documentação oficial.

O custo de se efetuar uma mudança é avaliado por rastreabilidade e domínio do requisito. Esse custo é estimado e, em seguida, também é decidido se a alteração vai ou não prosseguir.

Figura 7. Oportunidade e custo de incorporações.



Fonte: Fernandes (2019).

Por fim, a implementação de mudanças é aplicada, quando necessárias, nos requisitos e no sistema. Assim, a documentação de requisitos deve ser reorganizada para que possa esperar por outras modificações, sem afetar as demais.

Sommerville (2011, p. 79) cita um ponto importante que acontece indevidamente e com frequência entre os desenvolvedores, que é a tentação de mudar o sistema e, em seguida, retrospectivamente, modificar o documento de requisitos. Esse procedimento deve ser evitado, por várias situações, sendo que as que mais se destacam é a implementação ficar defasada e a facilidade em esquecer de acrescentar informações nos documentos de requisitos.

Pressman (2011, p. 13) cita um ponto fundamental para o gerenciamento dos requisitos: a disponibilização da função de qualidade (QFD). Essa técnica traduz as necessidade do cliente para requisitos técnicos do software; concentra-se em maximizar a satisfação dos interessados e, para tanto, utiliza daquilo que é mais valioso para o cliente.

O QFD relaciona três tipos de necessidades:

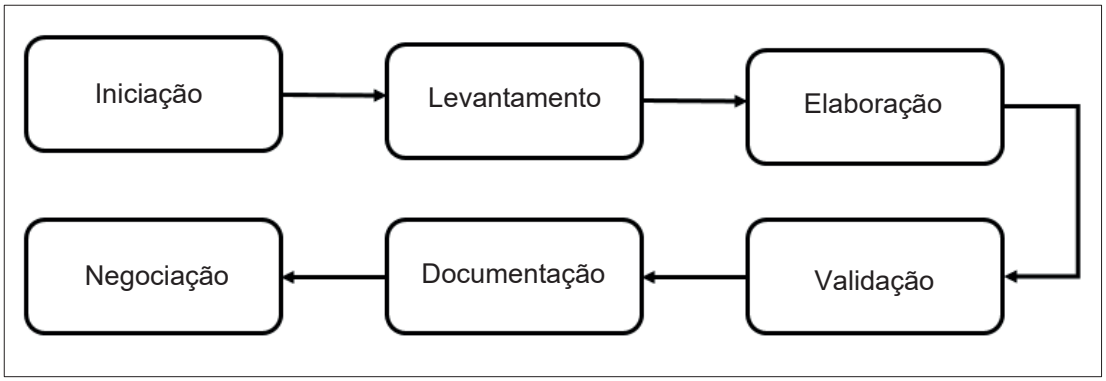
» **Requisitos normais**: refletem objetivos e metas estabelecidas para um produto durante reuniões com o cliente. Se o requisito estiver presente, o cliente fica satisfeito.

» **Requisitos esperados**: são requisitos que os clientes não declaram explicitamente, porém, se eles não estiverem presentes, o cliente ficará insatisfeito.

» **Requisitos fascinantes**: quando presentes, demonstram uma expectativa muito grande dos clientes.

As atividades de Gerenciamento de Requisitos, segundo Fernandes (2018, p. 92), podem ser divididas em:

Figura 8. Fases de atividades de gerenciamento de requisitos.



Fonte: Fernandes (2017 p. 92).

Na **iniciação**, o mais importante é a formalização e a necessidade de se criar um vínculo ou expectativa de negócio. O **levantamento de requisitos** trata a forma com que as técnicas se utilizarão para extrair requisitos dos usuários. Segundo Fernandes (2017 p. 92), a **elaboração de requisitos** se refere à reconciliação e tem o objetivo de analisar e classificar os requisitos levantados e ainda não tratados. Na **negociação de requisitos** o forte será a comunicação e a capacidade dos interessados resolverem conflitos de interesse ao projeto.

A **documentação de requisitos** irá refletir as necessidades, funcionalidades, concordâncias, restrições do sistema por um documento formal. O objetivo da **validação de requisitos** é garantir que o objetivo do sistema seja cumprido e é por isso que essa fase existe, para poder testar o sistema "no papel" para não ter prejuízos maiores após a implantação.

LEVANTAMENTOS DE REQUISITOS

UNIDADE III

CAPÍTULO 1

Conceito

Segundo Pressman (2001, p. 146), levantar requisitos é a combinação de elementos de resolução de problemas, elaboração, negociação e especificação. Sempre que possível, o gerente de projetos encorajará sua equipe para uma abordagem colaborativa, orientada a equipes em relação ao levantamento de requisitos. Os interessados deverão trabalhar juntos para levantar o problema, propor soluções preliminares, negociações para diferentes abordagens.

Coleta de requisitos

Segundo o PMBOK (2004, p. 430), o processo de coleta de requisitos determina, documenta e gerencia as necessidades das partes interessadas com a finalidade de atingir os objetivos do projeto e, seu principal benefício, é o fornecimento de uma base para definir e gerenciar o escopo do projeto, incluindo, o escopo do produto.

O envolvimento ativo das partes interessadas vai garantir o sucesso do projeto na descoberta e decomposição das necessidades em requisitos e na atenção tomada na determinação, documentação e gerenciamento dos requisitos do produto, serviço ou resultado do projeto. Também, além de incluir as necessidades qualificadas, documentadas, as expectativas do patrocinador e dos clientes, devem incluir condições e capacidade atendidas pelo projeto e pelo produto.

Por meio de uma EAP, é possível efetuar o planejamento do custo, cronograma e da qualidade. Os requisitos serão analisados, obtidos e registrados com os detalhes suficientes para serem inclusos, antes que o projeto se inicie, na linha de base do escopo.

Segundo o PMBOK (2004, p 112), o desenvolvimento dos requisitos começa com uma detalhada análise das informações contidas no termo de abertura do projeto, no registro e no plano de gerenciamento das partes interessadas. Muitas organizações agrupam os requisitos em vários tipos, tais como soluções de negócios e técnicas, sendo que a primeira se refere às necessidades das partes interessadas e a última a como essas necessidades serão implementadas.

Conforme vão sendo elaborados, os requisitos poderão ser agrupados por classificações que facilite um detalhamento e refinamentos posteriores:

- Necessidades de negócios: detalham as necessidade de um nível organizacional mais alto, como um todo, como oportunidades de negócios e os motivos que um projeto está sendo empreendido.
- Requisitos das partes interessadas: detalham as necessidades de uma parte ou de um grupo de partes interessadas.
- Requisitos de solução: detalha os atributos, as funções e as características do produto, serviço ou resultado que atenderão aos requisitos do negócio e das partes interessadas. Os requisitos de solução são ainda agrupados em requisitos funcionais e não funcionais.
- Requisitos de transição: definem as capacidades temporárias, tais como os requisitos de conversão de dados e de treinamento necessários à transição do estado atual de "como está" ao estado futuro de "como será".
- Os requisitos de projeto: determinam as ações, processos, ou outras condições que devem ser cumpridas pelo projeto.
- Os requisitos de qualidade: capturam quaisquer condições ou critérios necessários para validar a conclusão bem-sucedida de uma entrega de projeto ou o cumprimento de outros requisitos do projeto.

Matriz de rastreabilidade dos requisitos

A Matriz de Rastreabilidade de Requisitos é uma tabela que disponibilizará todas as ligações de requisitos de produto e suas origens, até que a entrega os satisfaça. Sua utilização ajuda garantir que cada requisito adiciona valor de negócio por meio da sua ligação aos objetivos de negócio e aos objetivos do

projeto. Ela fornece um meio de rastreamento do início ao fim do ciclo de vida do projeto, ajudando a garantir que os requisitos aprovados na documentação sejam entregues no final do projeto. Finalmente, ela fornece uma estrutura de gerenciamento das mudanças do escopo do produto.

Estão inclusos no rastreamento de requisitos:

» necessidades, oportunidades, metas e objetivos de negócio;

» objetivos do projeto;

» escopo do projeto/entregas da EAP;

» design do produto;

» desenvolvimento do produto;

» estratégia de teste e cenários de teste;

» requisitos de alto nível para requisitos mais detalhados.

Figura 9. Matriz de rastreabilidade de requisitos.

Matriz de rastreabilidade de requisitos

Nome do projeto:	
Centro de custo:	
Descrição do projeto:	

ID	ID Associado	Descrição dos requisitos	Necessidades do Negócio, metas e objetivos	Objetivos do projeto	Entregas de EAP	Design do Produto	Desenvolvimento do produto	Casos de teste
001	1.0							
	1.1							
	1.2							
	1.2.1							
002	2.0							
	2.0							
	2.1.1							
003	3.0							
	3.1							
	3.2							
004	4.0							
005	5.0							

Fonte: PMBOK (2004, p.118).

Os requisitos deverão ter seus atributos associados a cada registro da matriz de rastreabilidade de requisitos.

Segundo PMBOK (2004, p. 119), esses atributos auxiliam a definição de informações chave a respeito do requisito e podem incluir: um identificador único, uma descrição textual do requisito, os argumentos para sua inclusão, proprietário, fonte, prioridade, versão, status atual (se está ativo, cancelado, adiado, adicionado, aprovado, designado, concluído) e a data do status. Atributos adicionais para garantir que o requisito satisfaça às partes interessadas podem incluir estabilidade, complexidade e critérios de aceitação.

Problema de escopo

Os requisitos foram de forma precária ou não houve especificações suficientes, que, ao invés de esclarecer, criaram mais confusões.

Problema de entendimento

Ocorre quando o cliente ou usuário não estão certos do que precisam, têm um entendimento inadequado, ou ainda, problemas de ambiguidade.

Problema de volatilidade

Os requisitos vão mudando com o tempo, refletindo diretamente em custos e prazos, por causa dessas mudanças.

Segundo Pressman (2011, p. 132), na fase de levantamento de requisitos, é importante as partes interessadas estarem prontas para algumas perguntas iniciais e finais. As perguntas iniciais serão feitas na concepção do projeto e ajudará a identificar os interessados, os benefícios e as metas do projeto:

- Quem está por trás da solicitação do trabalho?
- Quem irá usar a solução?
- Qual será o benéficos econômico de uma solução bem-sucedidas?
- Há outra fonte para a solução que você precisa?

Já as perguntas finais concentrarão na eficácia das atividade de comunicação:

- Suas respostas serão oficiais?
- Minhas perguntas são relevantes para o seu problema?
- Deveria perguntar-lhe algo a mais?

CAPÍTULO 2
Identificação

As fontes de informação mais importantes de um processo de levantamento de requisitos são chamadas de partes interessadas. Elas devem ser identificadas para que o sistema possa ser desenvolvido de forma a atender corretamente as suas necessidades. Uma parte interessada, segundo Fernandes (2017, p.112), é alguém que pode ser materialmente afetado pela implementação de um sistema, sendo essa parte interessada também chamada de *stakeholder*, incluindo além de indivíduos, grupos de pessoas ou organizações.

O *stakeholder* pode ser aquele que irá utilizar o sistema, ser beneficiado ou prejudicado por ele ou ter algum tipo de responsabilidade com relação a ele. Normalmente, há muitas partes interessadas em um sistema. Para identificá-las, pode-se utilizar a definição de papéis ou cargos desempenhados na organização. Para realizar essa identificação, é possível perguntar ao cliente, analisar o organograma da organização e examinar o contexto do sistema.

A partir da correta identificação das partes interessadas envolvidas, deve-se apresentar a elas o papel que têm no processo de levantamento de requisitos, com suas características mais relevantes. É chegada a hora de identificar os usuários do sistema a ser desenvolvido, sendo usuário, por definição, todo aquele que irá interagir e operar o sistema quando ele estiver pronto e em funcionamento. Em um software, o usuário é aquele que está à frente do computador. Já em um outro exemplo, como um automóvel, os usuários podem ser o condutor e seus passageiros.

Cada usuário tem seu nível de responsabilidade e interação com o sistema. Os requisitos obtidos junto aos usuários que mais interagem com o sistema devem ter prioridade e maior relevância do que aqueles requisitos levantados junto aos que interagem menos. Alguns tipos de usuários também devem receber maior atenção aos detalhes de suas particularidades para garantir uma boa interação com o sistema. Dentre eles, podemos citar os portadores de deficiência, pessoas com níveis baixos de alfabetização, aqueles que não dominam os idiomas que serão utilizados no sistema e indivíduos com dificuldades no aprendizado e interação com sistemas de informática.

Além dos usuários definidos na fase de desenvolvimento do sistema, novos usuários podem ser identificados quando o sistema já estiver em utilização, para

fins de manutenção do sistema e promoção de eventuais melhorias e upgrades do sistema em novos ciclos de desenvolvimento.

Devemos nos atentar para o fato de que o desenvolvimento de um sistema sempre envolve custos. O cliente encomenda o sistema e é ele quem paga pelo seu desenvolvimento, sendo esta relação formalizada por um contrato, que também garantirá ao cliente que será fornecida toda a documentação técnica para a instalação e manutenção do sistema e, em caso de software, o seu código-fonte completo. Cabe ao cliente, normalmente gestores, donos de empresas ou responsáveis pelos departamentos nos quais o sistema será instalado, o poder de decisão sobre as funcionalidades do sistema e seus custos.

É de extrema importância que ocorra o apoio e o envolvimento contínuo do cliente durante o desenvolvimento do sistema, principalmente na fase de levantamento de requisitos para que a solução final de fato atenda às expectativas do cliente. O cliente é o comprador, o consumidor final, aquele que paga, logo, ele deve e precisa participar de todo o processo. Em alguns casos, quando o sistema é encomendado sob medida, os nomes dos compradores já são conhecidos antes mesmo do início do desenvolvimento.

No ramo de software, existem diversos formatos de negócio. Normalmente, oferecem a aquisição, a instalação e a utilização dos produtos, sendo que estes produtos podem ser desenvolvidos para depois serem vendidos aos compradores e disponibilizados para utilização dos usuários, dependendo da demanda e do mercado em específico.

Voltando ao levantamento de requisitos, existem algumas partes interessadas que podem ser envolvidas no processo de desenvolvimento do sistema, somente para contribuir e ajudar a identificar todos os requisitos importantes para o sistema. Tais partes interessadas são consideradas e identificadas como especialistas, pois têm conhecimentos, competências e experiências e elevado domínio sobre o assunto em questão.

Esses especialistas não precisam estar envolvidos com as tarefas técnicas do desenvolvimento, mas interagem com consultores e revisores, definindo requisitos, efetuando estudos sobre alguns aspectos do sistema em particular e embasando as tomadas de decisão da equipe de desenvolvimento.

Considerando as fases do processo em que estão mais envolvidos, os especialistas podem ser elementos da equipe de desenvolvimento. No ramo dos sistemas de

informação, podem ser analistas, programadores ou arquitetos de software. Já na indústria de construção civil, por exemplo, podem ser engenheiros civis e operários de construção.

Ainda podem existir as partes interessadas negativas, aqueles que desejam que o sistema não seja desenvolvido e nem esteja em funcionamento, podendo apresentar uma oposição pacífica ou uma hostilidade ativa. Identificar essas partes interessadas negativas é de extrema importância para que a equipe de projeto se proteja e fique atenta a possíveis tentativas de sabotagem ao sistema, normalmente, advindas de interesses pessoais e relações políticas em uma organização.

CAPÍTULO 3
Técnicas

Considerando os diversos tipos de organizações, projetos e tecnologias existentes, é imensa a variedade de métodos e técnicas que um engenheiro de projetos deve conhecer e dominar para um levantamento de requisitos bem executado.

Tais técnicas requerem do engenheiro uma soma de competências aplicadas em conjunto, com habilidades como:

- Questionar – saber realizar as perguntas certas sobre os requisitos para as pessoas certas.
- Observar – saber analisar corretamente o comportamento dos usuários de forma a identificar as reais necessidades deles com relação ao produto, sistema ou processo em questão.
- Discutir – debater sobre as necessidades com os usuários no intuito de facilitar o entendimento dos requisitos.
- Negociar – promover o consenso entre os usuários para decidir sobre os requisitos que devem ser incluídos, alterados ou excluídos.
- Supor – prever e antecipar funcionalidades que possam ser úteis para os usuários, principalmente na criação de novos produtos.

Obviamente, não é nada fácil encontrar um profissional que esteja neste nível de preparo para a realização do levantamento de requisitos. Isso acontece devido à escassa formação e especialização nesta área em específico. Segundo Fernandes (2017, p.120), a dificuldade está essencialmente em selecionar as técnicas adequadas e saber aplicá-las corretamente, considerando as características de cada projeto.

Existem, atualmente, várias categorias de técnicas propostas pela comunidade científica:

- Mercadologia – categoria em que existe um interesse com relação aos requisitos para que contribuam com o sucesso do sistema do ponto de vista comercial.

- Psicologia e sociologia – categoria em que o foco nos requisitos está na relação deles com os usuários, para que se promova a satisfação deles como indivíduos e agentes sociais.
- Concepção participativa – nesta categoria, a definição dos requisitos tem um envolvimento ativo dos usuários já que os sistemas afetam diretamente o trabalho desses usuários, descentralizando poderes e delegando funções dentro da organização.
- Fatores humanos e interação humano-máquina – neste, recebem atenção os requisitos de interface gráfica e as funcionalidades que facilitem a utilização do sistema para os usuários.
- Qualidade – o foco está na melhoria de um processo já existente, de forma que os requisitos possam garantir a qualidade do sistema para o usuário e para o cliente.
- Métodos formais – o objetivo é a precisão e o rigor matemático dos requisitos especificados.
- Análise orientada a objetos – especificamente para softwares e sistemas de informação, cujo interesse está na relação entre os requisitos e o processo de desenvolvimento do sistema, utilizando como base objetos reais.
- Análise estruturada – também para softwares e sistemas de informação, refere-se à relação entre os requisitos e o processo de desenvolvimento do sistema, porém, com base em processos funcionais e em dados já existentes;

> "O bom analista conhece muitas técnicas, mas também sabe quando deve usá-las ou não. Ele combina e modifica as técnicas de acordo com as necessidades específicas." Soren Lauesen (1942)

As técnicas de levantamento de requisitos são classificadas em três categorias conforme o tipo de fonte para captura dos requisitos. São elas:

1. Pessoas individualmente.
2. Equipes de pessoas.
3. Artefatos (que são documentos, leis, produtos e objetos).

A técnica mais utilizada na categoria de pessoas individualmente, sem dúvida alguma é a entrevista.

Entrevistas

Segundo o PMBOK (2004, p. 111), uma entrevista é um meio formal ou informal para se extrair informações das partes interessadas. Ela poderá ser feita por conversas diretas, perguntas preparadas ou espontâneas e do registro das respostas. As entrevistas são frequentemente conduzidas individualmente, entre um entrevistador e um entrevistado, mas podem envolver grupos de entrevistados e entrevistadores.

Segundo Sommerville (2011, p. 73), entrevistas formais ou informais com os stakeholders do sistema são parte da maioria dos processos de engenharia de requisitos. Nelas, os stakeholders são questionados sobre utilização dos sistemas prontos ou os que serão desenvolvidos, e podem ser de dois tipos:

- **Entrevistas fechadas**: com um conjunto de perguntas pré-definidas.
- **Entrevistas abertas**: as perguntas são exploradas para desenvolver uma melhor compreensão.

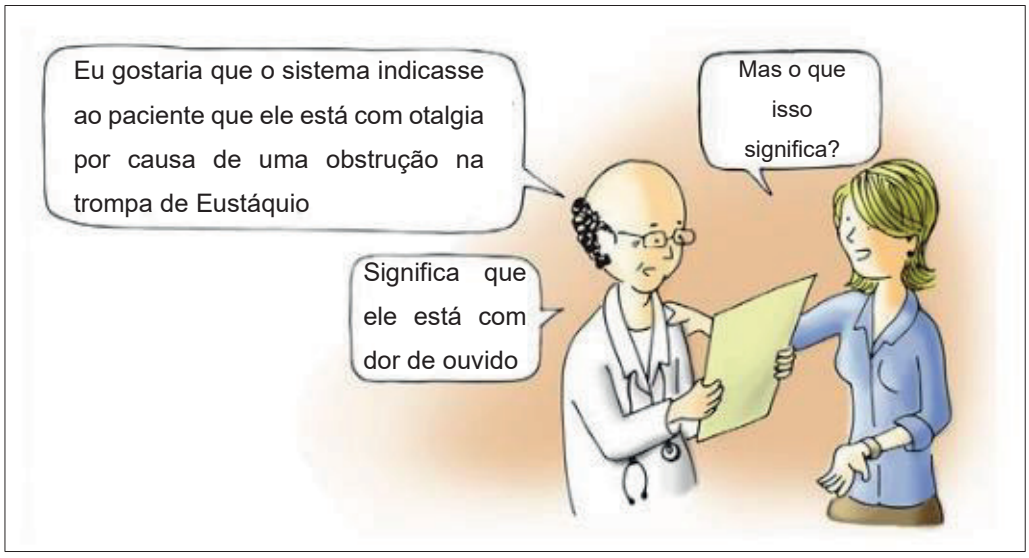
As entrevistas são uma mistura de ambas as práticas, abertas e fechadas. Algumas respostas para determinadas questões poderão nos levar a outras perguntas, tanto abertas como fechadas. Podemos ter a ideia de que discussões totalmente abertas sejam mais flexíveis de compreensão. Porém, o contexto da entrevista poderá nos levar a um caminho indesejado. As perguntas abertas podem abrir precedência para que os stakeholders falem de sua vida pessoal ou de particularidades do trabalho, pois, existe uma empolgação que as pessoas gostam de se envolver em entrevistas.

Geralmente, em certos projetos, a entrevista será a única fonte de informação das quais os entrevistadores terão acesso para a elaboração dos requisitos. O grande problema que dispondo somente da entrevista, poderá deixar faltar alguma informação essencial. Por isso, ela deve ser utilizada sempre com outras técnicas para a descoberta de requisitos.

Existem algumas dificuldades para descobrir conhecimento de domínio por meio de entrevistas. A primeira delas é a forma com que os especialistas usam seus jargões e terminologias diferentes para o mesmo domínio. Normalmente, eles utilizarão esses jargões e terminologias de acordo com seu dia a dia do trabalho, de forma mais precisa e sutil. Seria como o desenvolvedor fosse falar em termos técnicos para os usuários do sistema.

Outra situação que dificulta a elicitação do domínio é que o conhecimento para os especialistas é tão corriqueiro que, para explicá-lo, terão dificuldades, ou o pior, vão achar que é algo tão simples que os desenvolvedores já deverão saber. Sommerville (2011, p. 74) cita um exemplo de elicitação de requisitos envolvendo um bibliotecário experiente. Para ele, não é necessário dizer que todas as aquisições são catalogadas antes de serem adicionadas à biblioteca. No entanto, isso pode não ser óbvio para o entrevistador, e acaba não sendo levado em conta nos requisitos.

Figura 10. Uso de jargão e termos técnicos.



Fonte: Fernandes (2017, p. 100).

Ainda acerca das eficácias das entrevistas, ela pode não ser muito eficaz também quando o assunto envolve ambientes com restrições organizacionais, pois, entre as pessoas de uma empresa poderá haver pequenas relações de poder. Segundo Sommerville (2011, p. 73), as estruturas organizacionais publicadas raramente correspondem à realidade do processo de tomada de decisão em uma organização, mas os entrevistados podem preferir não revelar a estrutura real e sim a estrutura teórica, para um estranho. Na maioria das vezes, os entrevistados dificultam a discussão política e econômica organizacionais que poderão fazer falta para a elicitação dos requisitos.

Outro ponto fundamental para uma entrevista é o papel do entrevistador. Eles deverão ser muito versáteis e estarem sempre abertos para novas ideias, ouvir mais os stakeholders antes de formular uma solução ou transcrição. Também devem estimular os entrevistados a participar de discussões das quais o entrevistador os conduza a liberar informações pertinentes e relevantes aos requisitos, ou seja, o entrevistador deverá ser flexível para o novo, porém deve manter o contexto do domínio do sistema.

O fracasso ou o sucesso de uma entrevista dependerá do resultado da elaboração dos requisitos. Se algo faltar, poderá comprometer totalmente o projeto e, além de desagradar o cliente com um produto fora de escopo, poderá atrasar o cronograma ou mesmo não entregar o produto, causando prejuízos incalculáveis.

Uma vantagem que poderá acrescentar muito na elicitação de requisitos, utilizando as entrevistas, é entrevistar pessoas experientes, executivos envolvidos com informações confidenciais da empresa ou especialistas que poderão auxiliar em funcionalidades das quais os usuários "comuns" não teriam acesso. Isso normalmente está ligado ao envolvimento organizacional citado anteriormente.

Grupos de discussão

Reunir um grupo de discussão é selecionar pessoas que são representantes de clientes ou usuários de um produto para obter *feedback*. O feedback pode ser coletado sobre oportunidades, necessidades e problemas para determinar requisitos ou pode ser coletado para refinar e validar os requisitos já elicitados.

Segundo Sommerville (2011, p.112), os grupos de discussão reúnem as partes interessadas pré-qualificadas e os especialistas no assunto para aprender a respeito das suas expectativas e atitudes em relação a um produto, serviço ou resultados propostos. Um moderador treinado guia o grupo por uma discussão interativa, planejada para ser mais informal do que uma entrevista individual.

Esse tipo de pesquisa de mercado é diferente do brainstorming, no qual é um processo gerenciado com participantes específicos. Existe o risco de seguir a multidão e algumas pessoas pensam que os grupos de discussão são, na melhor das hipóteses, improdutivos.

Oficinas facilitadas

Para definir os requisitos de um sistema ou produto, são realizadas sessões focadas que reúnem as partes interessadas. Essas sessões são chamadas de Oficinas Facilitada, uma das técnicas primárias do levantamento de requisitos. Nelas, são estabelecidos os requisitos mais importantes e multifuncionais, de

forma rápida, e garantindo que sejam sanadas possíveis divergências entre as partes interessadas.

O maior intuito dessas sessões facilitadas é o de gerar confiança, além de promover a formação de um grupo com bom relacionamento, que interaja, troque informações e aprimore sua comunicação constantemente. Por meio dessas oficinas, questões são resolvidas mais rapidamente do que em sessões individuais.

Técnicas para criatividade em grupo

As atividades em grupo devem ser organizadas de modo a promover a identificação precisa dos requisitos do projeto, sistema ou produto.

Para que isso ocorra, são utilizadas algumas técnicas que desenvolvem a criatividade em grupo. São elas:

- **Brainstorming**: utiliza-se para gerar ideias diversas relacionadas ao projeto ou produto. Esta técnica funciona como uma primeira coleta de requisitos que em outras técnicas serão votados, escolhidos e priorizados.

- **Grupo Nominal**: amplia o brainstorming com um procedimento de votação para selecionar e organizar as melhores ideias e levá-las à priorização.

- **Mapas mentais**: as ideias selecionadas são consolidadas em um mapa mental que demonstra os atributos em comum e as divergências de entendimento, além de promover o surgimento de novas ideias.

- **Diagrama de afinidade**: permite classificar as ideias em grupos, assim fica mais fácil revisá-las e analisá-las.

- **Análise de decisão para critérios múltiplos**: utiliza uma matriz de decisão para uma abordagem analítica e sistemática com foco em definir critérios para avaliar e classificar muitas ideias conforme seus níveis de risco e incertezas.

Técnicas para tomadas de decisão em grupo

São técnicas que promovem a avaliação de múltiplas alternativas visando resultados que são as ações futuras esperadas. Essas técnicas classificam e priorizam os requisitos de um projeto, sistema ou produto. São elas:

» **Unanimidade**: acontece quando a decisão alcançada reflete que todos concordam com um único curso de ação. Uma forma de obter a unanimidade é por meio da técnica de Delphi, em que especialistas respondem alguns questionários e comentam sobre as respostas de cada coleta de requisitos de forma anônima.

» **Maioria**: decisão obtida com o apoio de mais de 50% dos membros do grupo. Um grupo com número ímpar de participantes garante que não haverá empate.

» **Pluralidade**: mesmo que a maioria não seja alcançada, a decisão é tomada pelo maior bloco do grupo. Acontece quando o número de opções de escolha é maior que duas.

» **Ditadura**: um único indivíduo decide pelo grupo.

Questionários e pesquisas

São perguntas preparadas para coletar informações rapidamente de um grande número de respondentes. Facilita a coleta quando os indivíduos estão geograficamente espalhados e quando se deseja aplicar análise estatística e as perguntas poderão ser abertas ou fechadas. É recomendável que seja feito um esforço na elaboração das perguntas para que sejam o máximo possível de perguntas fechadas. As perguntas abertas poderão ser passíveis de interpretação, tanto na pessoa que está lendo a pergunta, quanto a que lerá a resposta. Por exemplo, se você precisar das especificações de hardware do o servidor que a aplicação será instalada:

Quadro 9. Questionário.

Perguntas/Sugestões	Descrição
1 - A empresa possui um computador com os componentes próprios para arquitetura de servidores?	( ) Sim ( ) Não

Perguntas/Sugestões	Descrição
1.2 - Se sim, quais são as especificações dos seguintes componentes?	» Processador: › Quantidade:_____ › Marca: _____ › Modelo: _____ › Velocidade:_____ » Memória: › Capacidade____ ( ) GB/TB » Disco Rígido: › Quantidade: _____ › Capacidade: _____ ( ) GB/TB cada › SSD? ( ) Sim ( ) Não
1.3 - Se não, a empresa deverá possuir um computador com as seguintes especificações para que o sistema não tenha suas funcionalidades comprometidas pelos próximos 5 anos:	» Processador: › Quantidade: 8 › Marca: Intel › Modelo: Xeon › Velocidade: 3.5GHz » Memória: › Capacidade 128GB » Disco Rígido: › Quantidade: 4 › Capacidade: 1TB cada › SSD

Fonte: autor.

Veja que a pergunta um já sugere que o usuário preencha corretamente as respostas, caso a resposta seja positiva, pois tem um exemplo na alternativa 1.3. Em caso de negação da pergunta a alternativa 1.3 já será a especificação para a aquisição de um novo equipamento.

Observações

É a forma de examinar indivíduos em seu ambiente, analisando como executam seu trabalho e/ou determinadas tarefas e processos. Indicada quando há dificuldade no relato dos requisitos ou quando, por algum motivo, há recusa em expressar os requisitos. Pode ser realizada por um observador externo ou até mesmo por um observador participante, que também executa o processo em observação para experimentação no intuito de descobrir requisitos ocultos.

Protótipos

Uma forma mais precisa de obter requisitos é por construção de um protótipo, um modelo funcional do produto a ser desenvolvido. Sendo o protótipo tangível,

fica mais fácil levantar requisitos sobre algo que as partes interessadas possam experimentar. Podem ser interativos, com modelos de tamanho natural, promovendo assim na prática a experiência do usuário e coleta de suas opiniões. Quando os feedbacks recebidos a partir do protótipo e a coleta de requisitos forem suficientes, poderá partir para a fase de concepção do produto final.

Ainda é possível adotar a técnica de *storyboarding*, na qual o protótipo exibe uma sequência de imagens e ilustrações que apresentam o produto. São usados *storyboardings* em projetos de cinema, propaganda e de desenvolvimento ágil de softwares, em que é possível mostrar os caminhos de navegação pelas páginas, telas ou interfaces de usuários.

Benckmarking

É a comparação entre práticas planejadas ou reais para gerar ideias de melhorias e avaliar desempenhos de processos e operações de organizações distintas.

Diagramas de contexto

São um exemplo de modelo de escopo, descrevendo visualmente o produto e como as pessoas e outros sistemas interagem com ele. Mostram de forma clara o sistema de negócios que envolve o produto, como processos, equipamentos, sistema computacional, de forma a evidenciar os agentes que promovem a entrada e que recebem a saída desse sistema de negócios que é o contexto do produto em questão.

Análise de documentos

Obtém-se os requisitos pela análise de documentação já existente, identificando informações relevantes que podem ser consideradas para os requisitos. Pode-se analisar qualquer tipo de documento que se refira ao produto, sistema ou projeto em questão, como por exemplo, planos de negócios, acordos, solicitações de propostas, fluxos de processos, dados lógicos, documentações de softwares, documentação de processos, registros de problemas ou questões políticas, leis, códigos, portarias, entre outros documentos que estiverem disponíveis.

CAPÍTULO 4
Negociação e priorização

Negociação de requisitos

Conflitos sempre acontecerão quando se trata de chagar a um consenso em relação a requisitos e o motivo sempre é a visão contraditória das partes interessadas. Um exemplo, utilizando o contexto do desenvolvimento de software, citado por Fernandes (2017, p. 153), é quando uma das partes interessadas pode exigir que a aplicação seja reiniciada, no caso de uma falha ser detectada, enquanto outra pode sugerir que a aplicação continue a funcionar; ou outro exemplo de um conflito é a qualidade de serviço requerida pelos usuários entrar em contradição com as restrições orçamentais de quem vai pagar a aplicação.

O PMBOK (2010 p. 282) já dá como certo que conflitos são inevitáveis em um ambiente de projeto. As origens de conflitos incluem recursos escassos, prioridades de cronograma e estilos de trabalho pessoais. As regras básicas da equipe, as normas do grupo e práticas sólidas de gerenciamento de projetos, como planejamento das comunicações e definição de papéis, reduzem a quantidade de conflitos.

Segundo Pressman (2011, p. 145), as melhores negociações buscam incansavelmente os resultados “ganha-ganha”, ou seja, todas as partes interessadas terão seus requisitos satisfatórios, como por exemplo, os clientes ganham obtendo o produto que foi requisitado, os desenvolvedores trabalham com controle de prazos e cronogramas bem nítidos, orçamentos reais e atingíveis.

Gerenciar conflitos em projetos é uma habilidade que todo gestor deve, obrigatoriamente, possuir. Quando se faz um gerenciamento de conflitos bem-sucedido, o resultado reflete diretamente na produtividade e no relacionamento positivo entre os envolvidos. Isso também reflete em casos quando as diferenças de opiniões podem agregar valor ao processo criativo e melhora o processo de tomadas de decisão. Quando um conflito não é bem gerenciado, o caos poderá tomar conta do projeto, as diferenças se tornam impeditivos para o processo criativo.

Em uma atuação eficiente, o gerente de projetos na resolução de conflitos fará com que a comunicação seja eficaz. Quando o conflito ocorre, o gerente de

projeto deverá utilizar conversas privadas, direta e colaborativa. Em caso de reincidência, procedimentos formais, ações disciplinares e alguma regra legal da empresa, poderá ser utilizada.

De acordo com o PMBOK (2004, p. 9), os fatores que influenciam os métodos de resolução de conflitos incluem:

» importância relativa e intensidade do conflito;

» pressão de prazo para resolver o conflito;

» posição assumida pelas pessoas envolvidas;

» motivação para resolver o conflito a longo ou curto prazo.

Ainda, seguindo as boas práticas do PMBOK (2004, p. 9), existem cinco técnicas gerais para resolver conflitos. Como cada uma delas tem o seu lugar e sua função, elas não são apresentadas em nenhuma ordem específica:

» **Retirar/evitar**: recuar de uma situação de conflito atual ou potencial, adiando a questão até estar mais bem preparado, ou ser resolvida por outros.

» **Suavizar/acomodar**: enfatizar as áreas de acordo e não as diferenças, renunciando à sua posição em favor das necessidades das outras pessoas para manter a harmonia e os relacionamentos.

» **Comprometer/reconciliar**: encontrar soluções que tragam algum grau de satisfação para todas as partes, a fim de alcançar uma solução temporária ou parcial para o conflito.

» **Forçar/direcionar**: forçar um ponto de vista às custas de outro; oferecer apenas soluções ganha/perde, geralmente aplicadas por uma posição de poder para resolver uma emergência.

» **Colaborar/resolver o problema**: incorporar diversos pontos de vista e opiniões com perspectivas diferentes; exige uma atitude cooperativa e um diálogo aberto que normalmente conduz ao consenso e ao comprometimento.

Em um conflito, podemos ter desde situações moderadas ou fáceis de solucionar até o caos completo. Normalmente, para estudo de diagnósticos, utilizamos sempre o pior caso como objeto de estudo, pois a coleta de informações é muito maior. Como atribuição de todo gerente de projetos é ter habilidades interpessoais na resolução

de conflitos, não é uma boa ideia, primeiramente classificá-los de modo a tomarmos diferentes decisões, para problemas específicos.

Segundo Fernandes (2017, p. 158), podemos utilizar duas técnicas para atribuir prioridades para a resolução de conflitos:

» Método de priorização: baseiam-se em outorgar valores a diferentes aspectos dos requisitos.

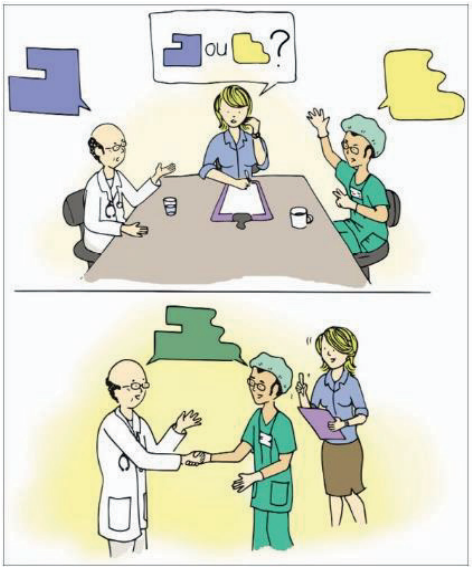
» Abordagens de negociação: concentram em atribuir prioridades aos requisitos após a concordância das partes interessadas.

Os processos de resolução de conflitos são bastante complexos, graças a naturais incompatibilidades e divergências entre várias partes interessadas.

Mas, de longe, a melhor forma de se resolver um conflito é a negociação. Para as partes envolvidas, as alternativas a serem negociadas, quando há, apresentarão interesses comuns e conflitantes. Muitos projetos de software falham, devido à ausência de negociação dos conflitos entre os requisitos. Estabelecer acordos entre as partes interessadas e conflitantes é difícil de alcançar.

Segundo Fernandes (2017, p. 153), a negociação é um processo social básico, utilizado para resolver conflitos, quem tem em sua essência, um processo de tomada de decisão aplicado em um contexto de interação estratégica ou de interdependência. Nas negociações, os participantes recorrem à informação e ao poder, para tentar influenciar o comportamento dos demais participantes. A execução de uma negociação, em um contexto de conflito entre requisitos, procura identificar uma entre várias alternativas aceitáveis e tecnologicamente viáveis.

Figura 11. A negociação como técnica para resolver conflitos.

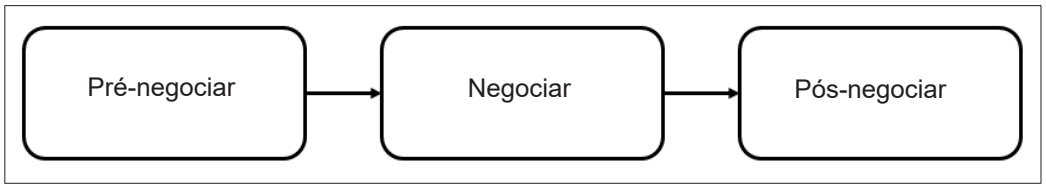


Fonte: Fernandes (2017, p. 153).

Processo de negociação

O sucesso nas técnicas de negociação é guaiado especialmente por três etapas:

Figura 12. Passos para processos de negociação.



Fonte: Fernandes (2017, p.153).

A pré-negociação foca como a negociação prosseguirá. Isso inclui as identificações das partes interessadas, os objetivos de cada etapa e resolução de possíveis conflitos, que serão caracterizados para que exista mais clareza em sua forma de resolução. Em seguida, deve-se montar as equipes com as pessoas habilitadas para esse processo.

A negociação executará todos os processos necessários para resoluções de conflitos, chegando a um acordo bom para todos, ou seja, mutualmente benéficas. Segundo Fernandes (2017, p. 154), uma negociação poderá ser resolvida por duas vias: consenso ou unanimidade, é necessário a aceitação por todas as partes envolvidas. Ou maioria, em que prevalece a vontade de um grupo contendo mais pessoas. Suas principais vantagens e desvantagens são:

Quadro 10. Comparação entre consenso e maioria.

Consenso	Maioria
Fomenta a cooperação.	Fomenta a competição.
Todos interessados têm o mesmo poder de decisão.	O poder de decisão equivale à hierarquia e a popularidade.
As partes interessadas concordam ou discordam.	As partes interessadas votam contra, a favor ou se abstêm.
Todas as ideias são discutidas e exploradas.	A maioria decide quais ideias devem ser consideradas.

Fonte: Fernandes (2017, p. 154).

Posturas e estratégias na negociação

- Segundo Fernandes (2017, p. 156), um participante deve ter certas posturas perante a negociação e, de forma genéricas, são:
  - **Inação**: significa indiferença pelo resultado da negociação.
  - **Competição**: ênfase em fazer prevalecer os próprios privilégios.
  - **Acomodação**: dar mais atenção aos interesses dos outros aos próprios interesses.

› **Colaboração**: ênfase para encontrar soluções ganho-ganho.

› **Compromisso**: concessões para encontrar soluções razoáveis.

Priorização de requisitos

Em gerenciamento de projetos, é fundamental estabelecer quais serão as prioridades das tarefas. Segundo Fernandes (2017, p. 158), as tarefas de um projeto de requisitos devem ser selecionadas por um conjunto de requisitos e, em seguida, separá-los em subconjuntos que incluam os requisitos mais importantes. Esse processo, denominado priorização de requisitos é uma técnica que ajuda na identificação dos requisitos fundamentais e os ordenar por grau de importância.

Critérios de priorização

Critérios podem ser utilizados para priorizarmos os requisitos que foram selecionados para serem priorizados. Alguns deles são destacados por Fernandes (2017, p. 161): importância, urgência, utilidade, penalização, tempo, custo e risco. Entretanto, não é nada produtivo associarmos uma diversidade de critérios e cada uma deve ser atribuído para determinada situação específica.

Técnicas de priorização

Segundo Fernandes (2011, p. 162), é muito importante associarmos um valor de atribuição de prioridades, a cada requisito considerado, pois isso estabelece uma ordem entre eles:

» **Top-10**: os dez requisitos mais importantes do projeto.

» **Classificação**: escala ordinal que os requisitos mais importantes ficam nas primeiras posições.

» **Agrupamento**: agrupados por grupos de importância. Normalmente, críticos, desejáveis e opcionais.

» Árvore de **pesquisa binária**: a Árvore de Pesquisa Binária é um algoritmo que normalmente é usado na busca de informações e pode ser facilmente escalado para ser usado na priorização de muitos requisitos.

» **Jogo de planejamento**: é um recurso de programação extrema e é usado com os clientes para priorizar recursos com base em histórias. Essa é uma variação da Técnica de atribuição numérica, na qual o cliente distribui os requisitos em três grupos: “aqueles sem os quais o sistema não funcionará”; “aqueles que são menos essenciais, mas fornecem valor comercial significativo”; e “aqueles que seriam bom ter”.

É altamente recomendável a comparação de várias técnicas de priorização de requisitos para ajudar na seleção de uma técnica adequada. Alguns exemplos de critérios de avaliação são:

» existe uma definição clara entre as etapas ou etapas do método de priorização;

» a saída numérica do método de priorização exibe claramente as prioridades do cliente para todos os requisitos;

» o método teve uma exposição e análise consideráveis na comunidade de engenharia de requisitos;

» é necessário um número razoável de horas para executar adequadamente o método de priorização;

» os engenheiros de requisitos e as partes interessadas podem compreender completamente o método dentro de um período razoável.

Qualquer projeto com limitações de recursos deve estabelecer as prioridades relativas da solicitação de recursos, casos de uso ou requisitos funcionais. A priorização ajuda o gerente de projeto a planejar lançamentos em etapas, assim, tome decisões de compensação e responda a solicitações para adicionar mais funcionalidade.

A priorização geralmente se torna política e emocionalmente carregada, com todas as partes interessadas, insistindo que suas necessidades são mais importantes. Uma abordagem melhor é basear as prioridades em alguns análise objetiva de como entregar o valor máximo do produto dentro do cronograma, do orçamento e das restrições de pessoal.

Uma abordagem mais analítica é avaliar o valor relativo do cliente e o custo relativo, risco técnico de cada recurso. O valor do cliente considera o benefício se um recurso estiver presente e a penalidade se não for. As características mais desejáveis devem ser aquelas que fornecem o maior valor ao menor custo ajustado ao risco.

MODELAGEM

UNIDADE IV

CAPÍTULO 1

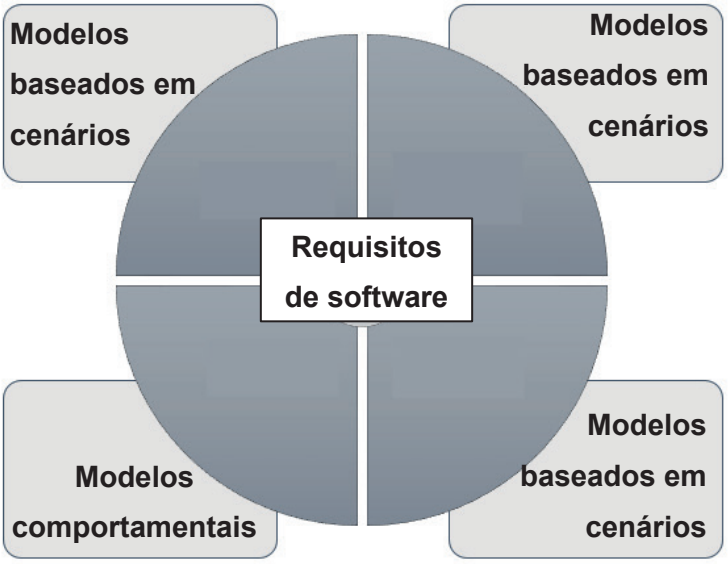
Definição de modelagem

Modelar um sistema significa desenvolver modelos abstratos em que cada modelo representa uma visão, um contexto, um domínio ou uma perspectiva do sistema. Ela normalmente representa o sistema com notações gráficas. A notação gráfica mais utilizada para representar um sistema, na fase de modelagem de requisitos é a UML.

Segundo Pressman (2011, p. 151), a ação da modelagem de requisitos resulta em um ou mais dos seguintes modelos:

- Modelos baseados em cenários de requisitos do ponto de vista de vários atores do sistema.
- Modelos de dados que representam o domínio de informações para o problema.
- Modelos orientados a classes que representam classes orientadas a objetos.
- Modelos orientado a fluxo que representam os elementos funcionais.
- Modelos comportamentais que representam como o software se comporta e eventos externos.

Figura 13. Elementos de um modelo de análise.

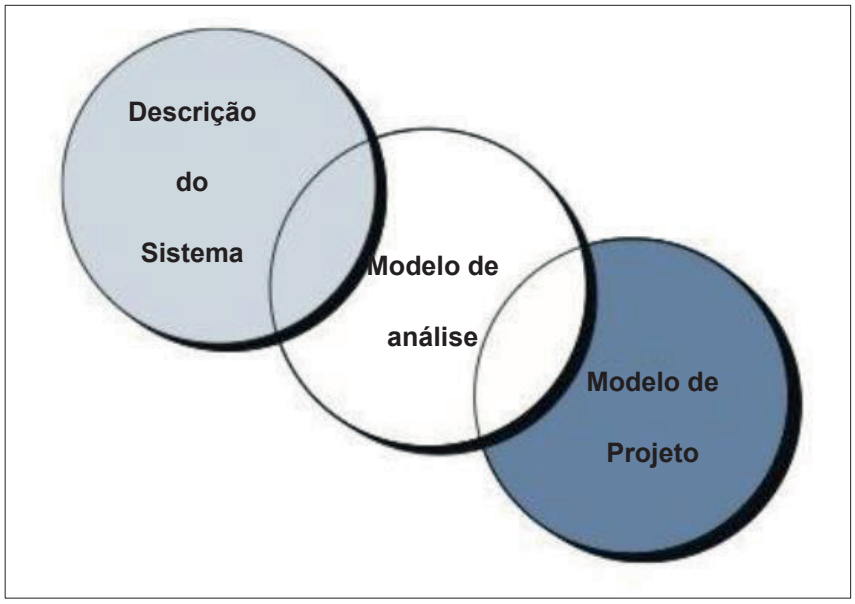


Fonte: Pressman (2011, p. 155).

Fernandes (2017, p. 205) define modelagem como o processo de obtenção de modelos sendo uma forma de transitar das ideias para os modelos, ela inclui um algum tipo de criatividade, não sendo um processo puramente técnico.

Esses modelos auxiliam tanto desenvolvedores, clientes e projetistas. Os desenvolvedores e clientes podem ver a qualidade logo após a conclusão do software. Já os projetistas recebem informações que podem ser traduzidas em arquiteturas, interfaces e componentes.

Figura 14. Modelo de requisitos, a descrição do sistema e o modelo de projeto.



Fonte: Pressman (2011, p. 151).

Segundo Pressman (2011, p. 152), o modelo de requisitos preenche a lacuna entre uma descrição sistêmica que descreve o sistema como um todo ou funcionalidade de negócios e deve alcança três objetivos: descrever o que o cliente solicita, estabelecer uma base para a criação de um projeto de software e definir um conjunto de requisitos que possa ser validado assim que o software for construído.

Sommerville (2011, p. 83) menciona que os modelos de requisitos são utilizados em momentos como durante a engenharia de requisitos para extrair os requisitos do sistema. Durante o processo de projeto, descrevem o sistema para a implementação e, depois, são utilizados para gerar a documentação de estrutura e operação do sistema.

Os modelos deverão atender tanto os sistemas existentes, auxiliando a discussão sobre pontos fortes e pontos fracos. Esses levam para os requisitos de sistema. Para novos sistemas, ajudam a explicar os requisitos propostos para *stakeholders*, os engenheiros para discutir a proposta de projetos e documentar implementações. Em um processo de engenharia, em que é dirigido a modelos, implementações completas ou parciais poderão ser geradas.

Segundo Ramos (2018), existem três formas principais de desenvolver modelos visuais, mais especificamente para apps, webapps e aplicações web. São eles:

- **Wireframe**: é a primeira demonstração visual do produto e é desenvolvida com lápis e papel e sua representação são esboços de baixa representação visual, baixa fidelidade e preocupa-se com a demonstração de todos os componentes, da maneira mais simples possíveis, porém, muito organizada e limpa.
- **Protótipo**: sempre confundido com o wireframe, é usado para simular o ambiente com o usuário. Por serem primeiro contato com o usuário, permite obter um feedback em testes de usabilidade. Muitas vezes, são mais rústicos do que os wireframe, porém, o resultado é mais interativo.
- **Mockup**: o mokup representa a finalização de um design por ser idêntico ao produto final. Bem feito, deve ajudar o usuário visualizar a estruturação da informação, visualização do conteúdo, visual definido e demonstração da forma de forma estática.

Quadro 11. Comparativos.

Comparativos		
Wireframe	Protótipo	Mockup
Estático	Estático	Estático
Representação Visual Baixa	Representação Visual Média	Representação Visual Alta
Comunicação rápida, fácil de se fazer.	Ótimo para teste de usabilidade	Representa quase que fielmente o visual final do produto. Muitas vezes utiliza para vendas

Fonte: Ramos (2018).

CAPÍTULO 2
Ontologia

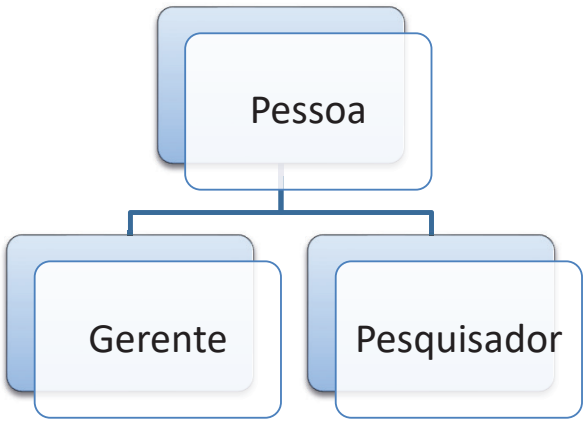
Segundo Fernandes (2017, p. 213), Ontologia é uma representação explicita dos conceitos importantes de um dado domínio, o conceito é uma categoria fundamental da existência e representação mental e uma classe de coisas.

Segundo Stabb (2009, p. 1), a palavra "ontologia" é usada com diferentes significados em diferentes comunidades. Pode ser utilizado em disciplinas filosóficas que lida com natureza e a estrutura da realidade; em metafísica, como a Ciência de "ser enquanto ser", estudo de atributos que pertencem às coisas ou a natureza deles.

As ontologias em Ciência da Computação são um artefato computacional utilizado para modelar formalmente a estrutura de um sistema de informação, seja entidades e relações relevantes que emergem de sua observação e que dão uteis para um propósito.

A espinha dorsal de uma ontologia consiste em uma hierarquia de generalização/ especialização. Imagine que estamos desenvolvendo requisitos para um sistema de recursos humanos. Pessoa, Gerente e Pesquisador serão conceitos importantes.

Figura 15. Estrutura hierárquica.



Fonte: autor.

Supondo que estamos interessados em aspectos relacionados aos recursos humanos, então Pessoa, Gerente e Pesquisador podem ser conceitos relevantes, sendo a Pessoa um super-conceito de Gerente e Pesquisador. A cooperação entre eles será relevante e uma pessoa concreta, que trabalha em uma empresa seria então uma instância de seu conceito direto.

Segundo Fernandes (2017, p. 214), as ontologias podem ser representadas por mapas conceituais que usam conceitos, normalmente representados dentro de um retângulo e, relações entre pares de conceitos, expresso por uma linha e frases de ligação, que define essa relação.

Em Sommerville (2011, p. 19), as ontologias são uma forma de padronizar as formas como a terminologia é usada e definir as relações entre diferentes termos. Elas têm sido cada vez mais usadas para ajudar a atribuir semântica para descrições em linguagem natural. Um exemplo de ontologias foi uma representação em Webservices de uma linguagem chamada OWL-S:

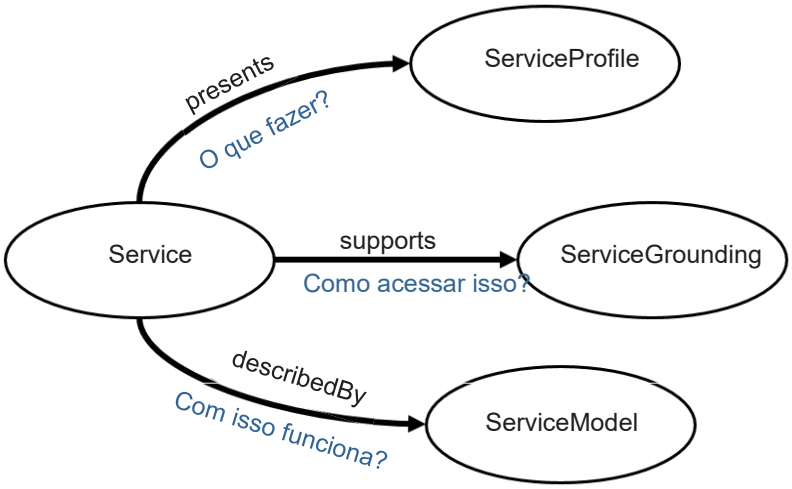
Uma ontologia superior para serviços

Veja um exemplo encontrado em https://www.w3.org/Submission/OWL-S/, acessado em 25 de setembro de 2019. A estruturação da ontologia de serviços é motivada pela necessidade de fornecer três tipos essenciais de conhecimento sobre um serviço, cada um caracterizado pela pergunta que ele responde.

A classe Serviço fornece um ponto de referência organizacional para um serviço Web declarado e uma instância do Serviço existirá para cada serviço publicado distinto. As propriedades *presents*, *describedBy* e *supports* são propriedades do Serviço. As classes *ServiceProfile*, *ServiceModel* e *ServiceGrounding* são os respectivos intervalos dessas propriedades.

De um modo geral, o *ServiceProfile* fornece as informações necessárias para um agente descobrir um serviço, enquanto o *ServiceModel* e o *ServiceGrounding*, juntos, fornecem informações suficientes para que um agente faça uso de um serviço, uma vez encontrado.

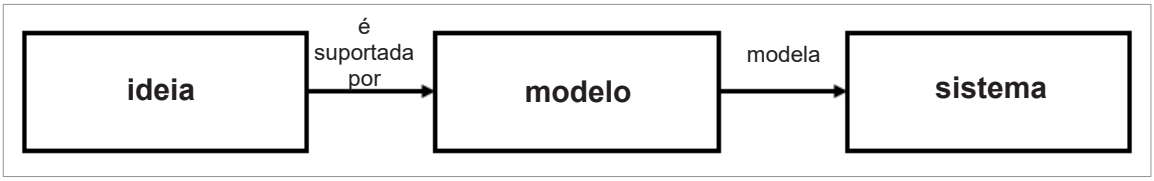
Figura 16. Nível superior da ontologia de serviço.



Fonte: https://www.w3.org/Submission/OWL-S/Service-Ontology1.1.gif.

Os conceitos sistema e modelo estabelecem a primeira relação relevante da ontologia, pois "um modelo modela um sistema" e, quando se estabelece uma relação unidirecional entre eles, existe igualmente uma relação inversa, além de o sistema ser um primeiro esforço de formalização em relação ao sistema alvo, e sua existência ocorre no âmbito mental, abrangendo a visão, as necessidades e as expectativas. (FERNANDES, 2019, p. 214).

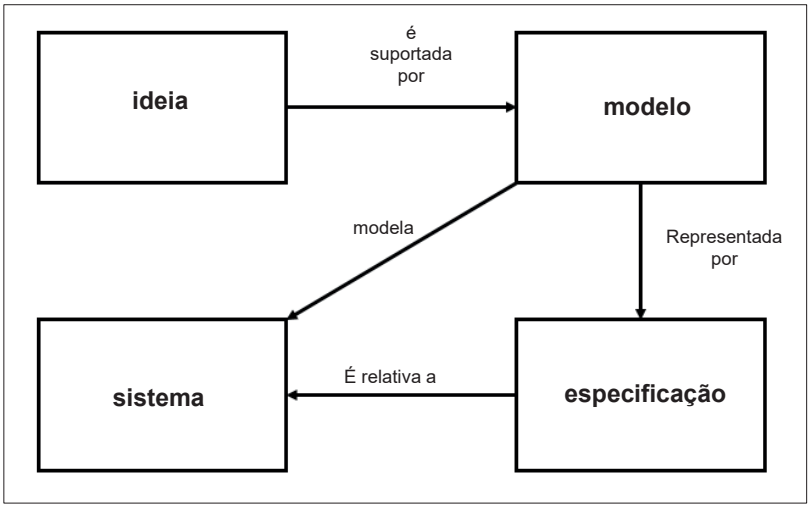
Figura 17. Um modelo está relacionado a um sistema.



Fonte: Fernandes (2017, p. 214).

Em modelagem, a especificação é uma representação real do modelo que foi escrito. Ela acaba sendo tangível e visível do modelo.

Figura 18. Ontologia: um modelo de especificação.



Fonte: Fernandes (2017, p. 216).

Segundo Fernandes (2017, p. 219), o diagrama é um contexto importante para a modelagem de software, pois, essa representação gráfica, visual de um conjunto lógico de elementos inter-relacionados, representa a expressão dos símbolos habitualmente dispostos em um espaço bidimensional, expressando um modelo.

Podemos ainda ter modelos representados por linguagens que deverão possuir definições e notações adequadas, baseadas em semântica e sintaxes. Uma notação é um conjunto de símbolos e sinais utilizados para a construção de representações, conceitos e as relações dos domínios que se aplicam no sistema em questão (FERNANDES, 2017, p. 219).

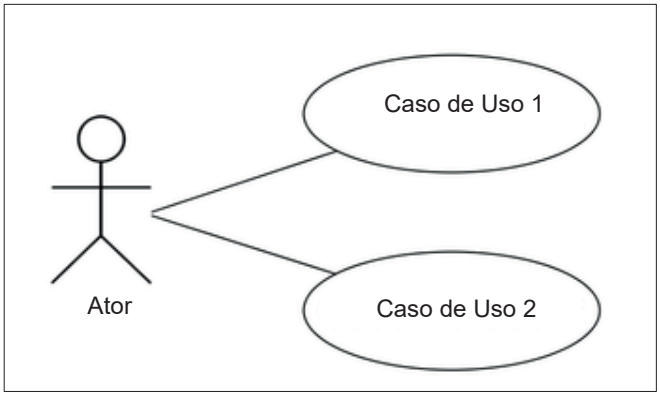
Durante o processo de desenvolvimento de software, desenvolvedores e clientes normalmente discutem até chegar a um acordo sobre requisitos que especificam a funcionalidade de um sistema que está sendo desenvolvido. Tais requisitos desempenham um papel crucial no ciclo de vida de desenvolvimento, já que eles formam a base para implementações, planos de trabalho, estimativas de custos e acompanhamento de diretivas correspondente. Em geral, os requisitos de software podem ser expressos de várias maneiras diferentes, incluindo o uso de diagramas de UML e estórias, porém, os mais usuais são expressos em linguagem natural, como por exemplo:

» (1.2) Um usuário deve ser capaz de fazer login no sistema: enquanto os requisitos expressos em linguagem natural têm a vantagem de ser legível para ambos os clientes e desenvolvedores, eles podem se tornar ambíguo, vago e incompleto. Embora linguagens formais possam ser usadas como uma alternativa que elimina alguns desses problemas, os clientes raramente são equipados com o conhecimento matemático e técnico para entender os requisitos altamente formalizados.

Os símbolos disponíveis em uma linguagem definem os componentes do domínio, a sintaxe define as estruturas de como representar e a semântica define a interpretação de cada símbolo e de como interpretá-lo no contexto do domínio em questão.

Segundo Fernandes, a expressão material de um modelo deve levar em consideração e forçar um caminho em direção a alguns sentidos humanos, como visão, audição ou tato. Os modelos de software normalmente são representados por linguagens textuais, como *Java, Ruby, Python*. As linguagens visuais são representadas por meio de um conjunto de diagramas, como o padrão que o mercado de desenvolvimento mais usa, a UML.

Figura 19. Exemplo de UML.



Fonte: autor.

CAPÍTULO 3
Modelos para requisitos

Modelos de domínio

A modelagem de domínio é uma maneira de descrever e modelar entidades do mundo real e os relacionamentos entre elas, que descrevem coletivamente o espaço no domínio do problema.

Em um nível básico, a modelagem de domínio pode ser entendida com "domínio" como a soma total do conhecimento do negócio e "modelagem" como uma abstração orientada a objetos da lógica de negócios.

Segundo Fernandes (2017, p. 227), um modelo de domínio constitui uma descrição das propriedades comuns de um determinado domínio, do sistema que está sendo desenvolvido. Sua grande vantagem é a possibilidade de descrever e restringir o âmbito do domínio do problema.

A natureza baseada em objeto da modelagem de domínio pode ajudar o arquiteto a controlar o desenvolvimento de um aplicativo mais facilmente. Ele insiste na coesão e reutilização dos objetos e encapsula a lógica de negócios de maneira mais intuitiva.

Modelos de caso de uso

Um caso de uso representa uma tarefa e sua forma é mostrado como uma elipse e atores como "bonecos" de figura de palitos. Por dar uma visão simplista, o diagrama de caso de uso precisa oferecer mais detalhes para o entendimento, podendo ser uma simples descrição textual, descrição estruturada ou tabela, como sugere Sommerville (2011, p. 86)

Figura 20. Caso de uso de transferência de dados.

Fonte: Sommerville (2011, p. 86).

O quadro sugerido com as descrições que complementam um diagrama de caso de uso é:

Quadro 12. Descrição complementar do diagrama de caso de uso.

Atores	Recepcionista do médico, sistema de registros de pacientes
Descrição	Uma recepcionista pode transferir dados para um banco de dados. As informações transferidas podem ser atualizadas com as informações pessoais (endereço, telefone etc.) ou com um resumo do diagnóstico e tratamento do paciente.
Dados	Informações pessoais do paciente, resumo do tratamento.
Estímulos	Comando de usuário emitido pela recepcionista do médico.
Resposta	Confirmação de que o PRS foi atualizado.
Comentários	A recepcionista deve ter permissões de proteção adequadas para acessar as informações do paciente.

Fonte: Sommerville (2011, p. 87).

Diagramas de sequência

Segundo Sommerville (2011, p. 87), os diagramas de sequência em UML são usados, principalmente, para modelar as interações entre os atores e os objetos em um sistema e as interações entre os próprios objetos. A UML tem uma sintaxe rica para diagramas de sequência, que permite a modelagem de vários tipos de interação. Como não tenho espaço para cobrir todas as possibilidades aqui, concentro-me nos fundamentos desse tipo de diagrama.

O diagrama mostra a sequência de interações ocorridas durante um caso de uso qualquer ou em uma instância de caso de uso.

Veja no caso de uso a seguir, de Sommerville (2011, p. 88):

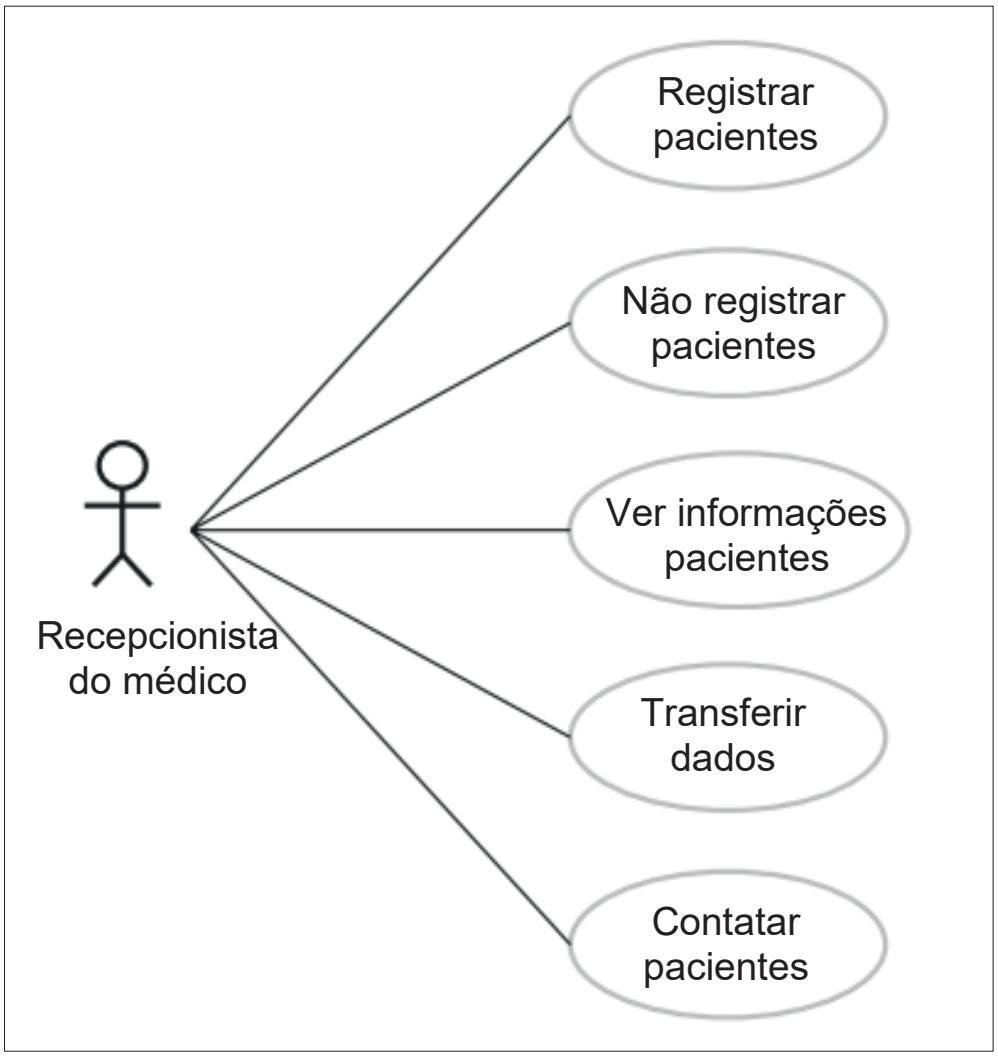
» A recepcionista do médico aciona o método *VerInfo()* em uma instância P da classe de objeto *InfoPaciente*, fornecendo o identificador do paciente PID (Paciente ID). P é um objeto de interface do usuário, exibido como um formulário que mostra os dados do paciente.

» A instância P chama o banco de dados para retornar as informações necessárias, fornecendo o identificador da recepcionista, que permite a verificação de proteção UID (User ID).

» O banco de dados verifica que o usuário está autorizado a essa ação.

» Se autorizado, as informações de pacientes são retornadas, e um formulário é preenchido na tela do usuário. Se falhar a autorização, aparece uma mensagem de erro.

Figura 21. Casos de uso envolvendo o papel da "recepcionista do médico".



Fonte: Sommerville (2011, p. 88).

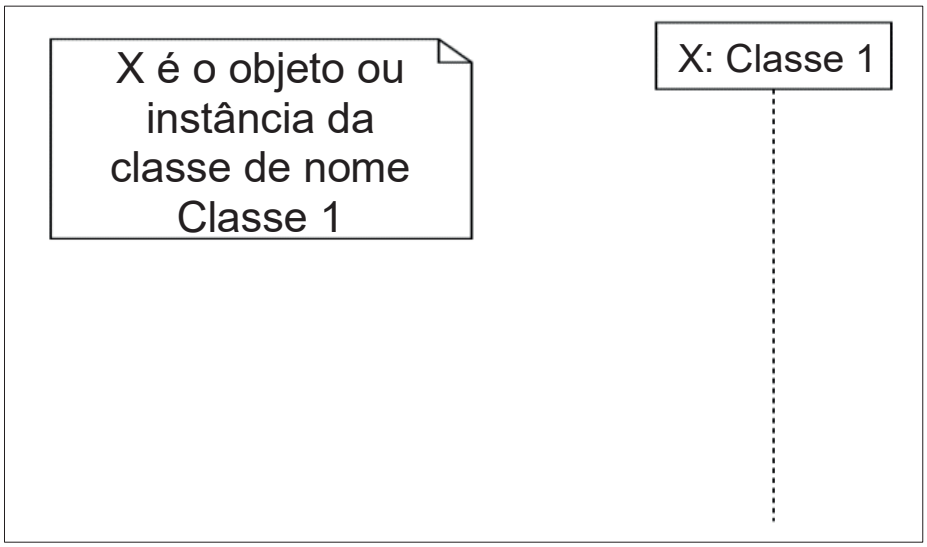
Após criado o caso de uso, inicia-se o desenho do diagrama de sequência. Ele é composto por atores na parte superior, com uma linha tracejada verticalmente. As interações entre os objetos são setas com suas anotações. Um retângulo na linha tracejada informa a linha da vida do objeto. A leitura do diagrama deve ser feita de cima para baixo.

Uma linha de vida é um elemento nomeado que representa um participante individual em um diagrama de sequência. Basicamente, cada instância em um diagrama de sequência é representada por uma linha de vida. Os elementos da linha de vida estão localizados na parte superior de um diagrama de sequência. O padrão de nomeação de uma linha da vida é: Nome da Instância -> Nome da Classe.

Exibimos uma linha de vida em um retângulo chamado *head* com seu nome e tipo. A *head* está localizada no topo de uma linha tracejada vertical (conhecida

como haste). Se queremos modelar uma instância sem nome, seguimos o mesmo padrão, exceto que agora a parte do nome da linha de vida é deixada em branco.

Figura 22. Linha da vida.



Fonte: autor.

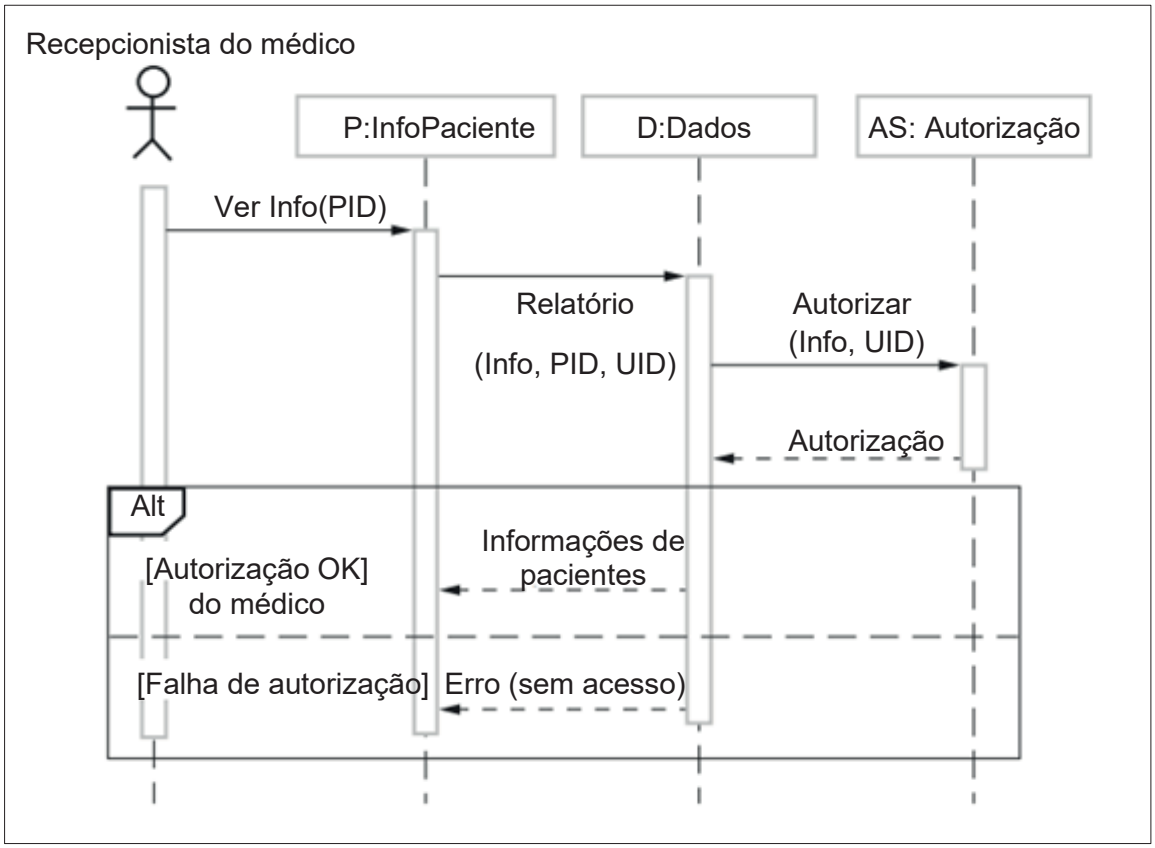
Mensagens – a comunicação entre objetos é representada usando mensagens. As mensagens aparecem em uma ordem sequencial na linha de vida. Representamos mensagens usando setas. Linhas de vida e mensagens formam o núcleo de um diagrama de sequência.

Mensagens síncronas – uma mensagem síncrona aguarda uma resposta antes que a interação possa avançar. O remetente aguarda até o destinatário concluir o processamento da mensagem. O chamador continua apenas quando sabe que o destinatário processou a mensagem anterior, ou seja, recebe uma mensagem de resposta. Grande número de chamadas na programação orientada a objetos é síncrono. Usamos uma ponta de seta sólida para representar uma mensagem síncrona.

Mensagens assíncronas – uma mensagem assíncrona não espera uma resposta do destinatário. A interação avança, independentemente de o receptor processar a mensagem anterior ou não. Usamos uma ponta de seta alinhada para representar uma mensagem assíncrona.

Veja na figura a seguir os conceitos que modelam as interações envolvidas no caso de uso e suas notações:

Figura 23. Diagrama de sequência para "Ver informações de pacientes".



Fonte: Sommerville (2011, p. 88).

Modelos comportamentais

Segundo Sommerville (2011, p. 93), são modelos do comportamento dinâmico do sistema quando está em execução e mostram o que acontece ou deve acontecer quando o sistema responde a um estímulo de seu ambiente:

- Modelagem orientada a dados – alguns dados que chegam precisam ser processados pelo sistema.
- Modelagem orientadas a eventos – alguns eventos que acontecem disparam o processamento do sistema.

Os sistemas de negócios são dirigidos a dados, são controlados pela entrada de dados com processamentos de eventos externos e envolvem sequências de ações nos dados e a geração de uma saída.

Usando o exemplo de um sistema de chamadas telefônicas, a ligação feita pelo cliente, aciona um estímulo do qual serão calculados os custos e gerará a conta para o cliente.

Modelagem orientada a dados

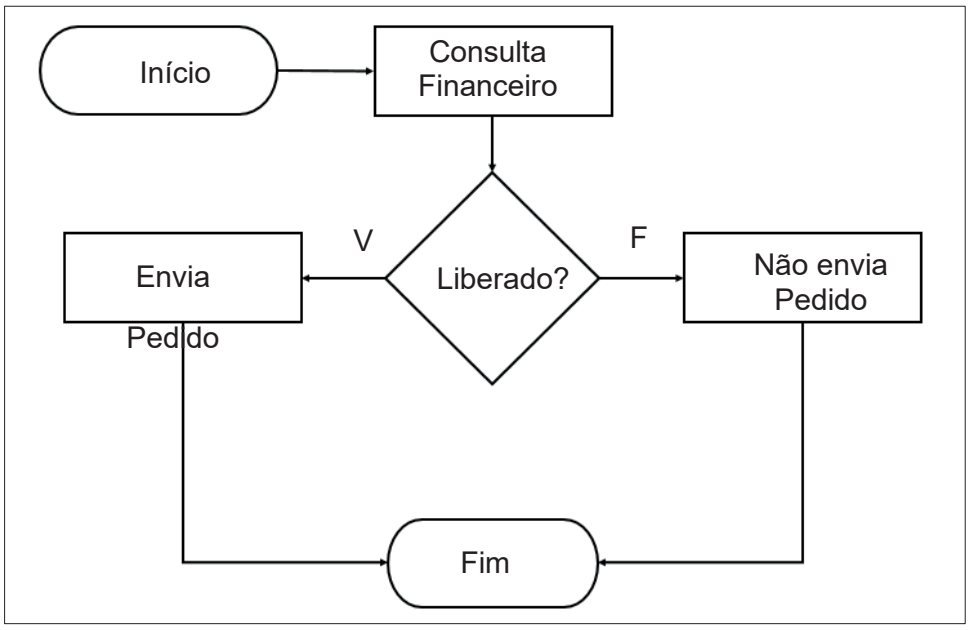
Segundo Sommerville (2011, p. 94), modelos dirigidos a dados mostram a sequência de ações envolvidas no processamento de dados de entrada e a geração de uma saída associada. Eles são particularmente úteis durante a análise de requisitos, pois podem ser usados para mostrar, do início ao fim, o processamento de um sistema.

Ou seja, eles mostram toda a sequência de ações, desde uma entrada sendo processada até a saída correspondente, que é a resposta do sistema.

Os modelos dirigidos a dados estavam entre os primeiros que foram criados. Já na década de 1970, modelos estruturados, os DFDs fizeram parte desse acontecimento.

Os DFDs (Diagramas de Fluxo de Dados) são úteis pois documentam a movimentação dos processos, além de serem muito fáceis de serem explicados aos usuários do sistema. Entretanto, a UML não oferece apoio ao DFD pois, segundo Sommerville (2011, p. 85), a razão para isso é que os DFDs se centram sobre as funções do sistema e não reconhecem os objetos do sistema.

Figura 24. Diagrama de fluxo de dados.



Fonte: autor.

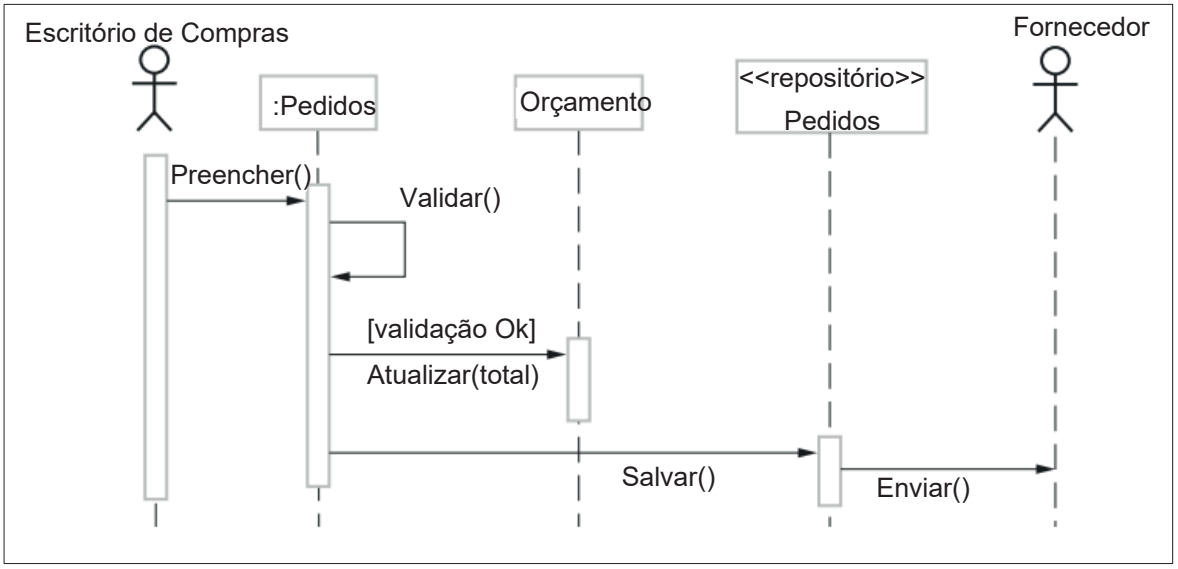
Devido aos sistemas dirigidos a dados serem tão comuns no mundo dos negócios, a UML 2.0 introduziu diagramas de atividades, semelhantes aos diagramas de fluxo de dados.

Modelagem dirigida a eventos

Segundo Sommerville (2011, p. 94), modelagem dirigida a eventos mostra como o sistema reage a eventos externos e internos. Ela é baseada na suposição de que um sistema tem um número finito de estados e que os eventos (estímulos) podem causar uma transição de um estado para outro. Essa percepção de um sistema é particularmente adequada para sistemas de tempo real. O problema com a modelagem baseada em estados é que o número de possíveis estados aumenta rapidamente. Para modelos de sistemas de grande porte, portanto, você precisa esconder detalhes nos modelos.

A UML apoia a modelagem baseada em eventos com diagramas de estado, pois eles mostram os estados do sistema e os eventos que causam transições de um estado para outro. Eles não mostram o fluxo de dados dentro do sistema, mas podem incluir informações adicionais sobre os processamentos realizados em cada estado.

Figura 25. Modelagem dirigida a eventos.



Fonte: Sommerville (2011, p. 94).

Método Jackson

O *Jackson System Development* (JSD) é um método de desenvolvimento de sistema que cobre o ciclo de vida do software diretamente ou fornecendo uma estrutura na qual as técnicas mais especializadas podem se encaixar. É utilizado para definir comportamento de entidade do mundo real por tempo, usando sequências de eventos.

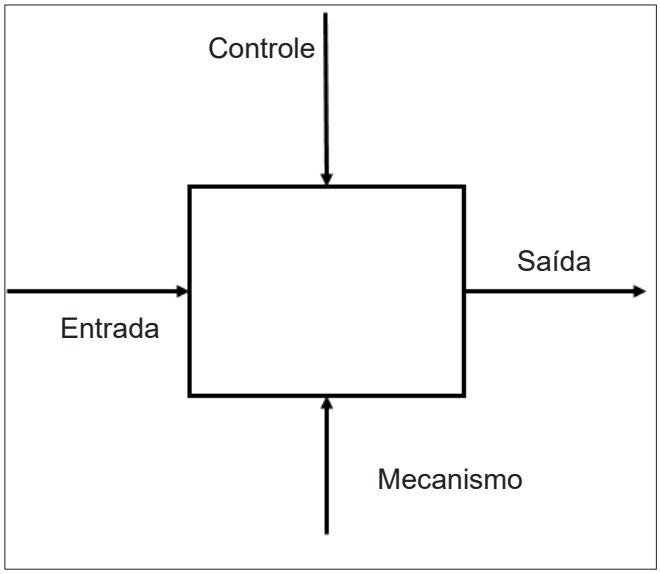
SADT

É uma metodologia de engenharia de sistemas e engenharia de software para descrever sistemas como uma hierarquia de funções. O SADT é uma linguagem de análise estruturada, que utiliza dois tipos de diagramas: modelos de atividades e modelos de dados. Foi desenvolvido no final dos anos 1960 por Douglas T. Ross e foi formalizado e publicado como IDEF0, em 1981.

As notações da SADT consistem em diagramas de setas de caixa (blocos), com quatro setas de cada lado definidas como: entrada, saída, controle e mecanismo e uma atividade no meio. Suas definições consistem no seguinte:

- Atividade: uma atividade é qualquer função ou processo que serve para transformar entradas em saídas.
- Entrada: os dados e as informações requeridos por uma atividade para iniciar o processo de transformação.
- Saída: os dados e as informações produzidos pela atividade como resultado dessa transformação.
- Controle: qualquer restrição que afeta o comportamento da atividade de alguma forma.
- Mecanismo: pessoas, recursos ou quaisquer meios necessários para executar a atividade.

Figura 26. Elemento de base SADT.



Fonte: autor.

Desenvolvimento conjunto de aplicativos (JAD)

O JAD foi projetado especificamente para o desenvolvimento de grandes sistemas de computadores. O objetivo do JAD é envolver todas as partes interessadas na fase de design do produto por meio de reuniões altamente estruturadas e focadas. Os participantes típicos da sessão incluem um facilitador, usuários finais do produto, principais desenvolvedores e observadores.

Nas fases preliminares do JAD, a equipe de engenharia de requisitos é incumbida de encontrar fatos e coletar informações. Normalmente, as saídas dessa fase, aplicadas à elicitação de requisitos de segurança, são objetivos e artefatos de segurança. A sessão JAD real é usada para validar essas informações, estabelecendo um conjunto acordado de requisitos de segurança para o produto.

Método de Requisitos Acelerados (ARM)

O processo ARM [Hubbard 00] é uma atividade de elicitação e descrição de requisitos facilitada. Existem três fases do processo:

- Fase de preparação.
- Fase de sessão facilitada.
- Fase de encerramento da entrega.

Durante a fase de preparação, o planejamento e a preparação são concluídos para garantir uma sessão eficaz. Durante esta atividade, são definidas as metas e os objetivos gerais, assim como o escopo preliminar do esforço; são definidas as principais medidas de sucesso; os principais participantes são identificados; e o cronograma preliminar é desenvolvido. A fase de preparação geralmente tem uma duração de um a quatro dias.

Durante a fase da sessão, um facilitador treinado e neutro em conteúdo conduz os participantes selecionados por meio de um processo estruturado para coletar os requisitos funcionais do projeto em consideração. O processo facilitado emprega técnicas definidas de escopo, debate de ideias e explicação e priorização. Esse estágio, geralmente, dura três dias.

Durante a fase de fechamento, as principais entregas, como a coleta de requisitos, são polidas, publicadas e disseminadas, e as várias atividades a seguir são planejadas.

O processo ARM é semelhante ao JAD, mas possui algumas diferenças significativas em relação ao método JAD da linha de base, o que contribui para sua exclusividade. Por exemplo, nesse processo, os facilitadores são neutros em conteúdo, as técnicas dinâmicas de grupo usadas são diferentes daquelas usadas no JAD, as técnicas de brainstorming usadas são diferentes e os requisitos são registrados e organizados usando diferentes modelos conceituais.

MACIASZEK, L. **Requirements Analysis and System Design**. Addison-Wesley, 2001. Esse livro concentra-se na análise de sistemas de informação e discute como diferentes modelos da UML podem ser usados no processo de análise.

MELLOR, S. J.; SCOTT, K.; WEISE, D. **MDA Distilled**: Principles of Model-driven Architecture. Addison-Wesley, 2004. Essa é uma introdução concisa e acessível para o método MDA. Foi escrita por entusiastas, de modo que o livro diz muito pouco sobre possíveis problemas com essa abordagem.

STEVENS, P.; POOLEY, R. **Using UML**: Software Engineering with Objects and Components. 2. ed. Addison-Wesley, 2006. Uma introdução curta e legível para o uso da UML na especificação e projeto de sistemas. Esse livro é excelente para aprender e entender a UML, embora não seja uma descrição completa da notação.

CAPÍTULO 4

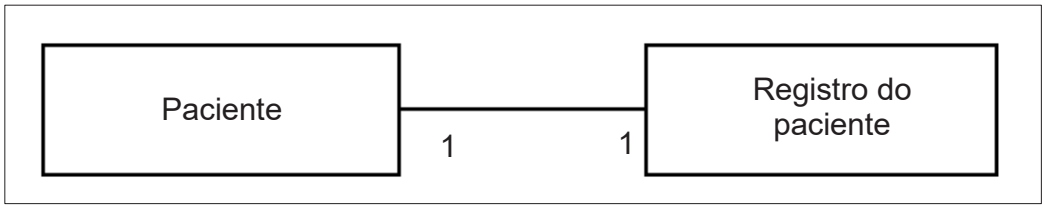
Modelos para requisitos orientados a objetos

Diagramas de classe

São utilizados em modelos de sistema orientados a objetos e demonstram suas classes e suas associações. A classe é uma definição de um tipo de objeto do sistema e uma associação é um link que denota um relacionamento entre as classes do diagrama. No desenvolvimento dos modelos, os objetos representam algo do mundo real, como paciente, receita médica, um médico etc. A aplicação desenvolvida, normalmente define objetos adicionais de implementação que dão funcionalidades requeridas do sistema.

Os diagramas de classe UML são expressos e podem ser demostrados em níveis de detalhamento diversos. No primeiro estágio, o detalhamento é o mais básico e identifica os objetos essenciais e a maneira mais fácil de fazer isso são os símbolos de classe. Se houver ligação, também deverá ser representada.

Figura 27. Classes e associação em UML.



Fonte: Sommerville (2011, p. 90).

Na figura 27, podemos representar um diagrama de classes bem simples, com duas classes: paciente e registro de paciente e uma associação entre eles.

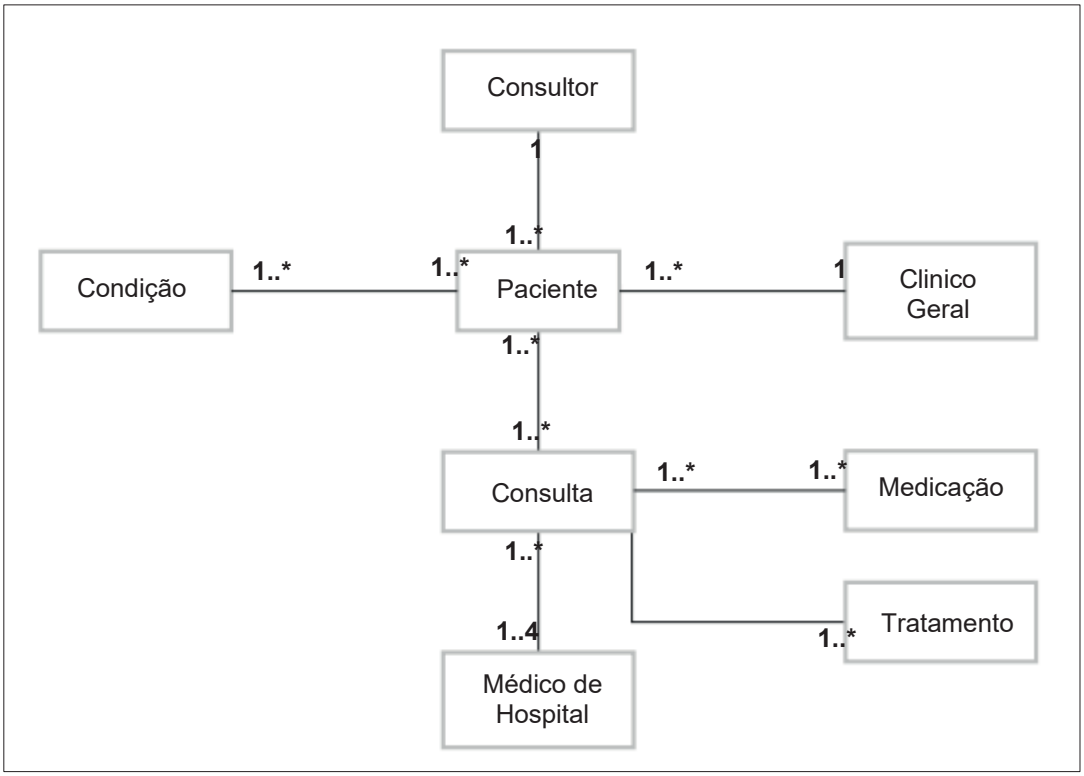
Na próxima figura, teremos outra característica do diagrama de classe: a capacidade de mostrar a quantidade de objetos envolvidos na associação. No diagrama, cada extremidade é anotada com “1”, que significa o relacionamento “1 para 1”. No exemplo de Sommerville (2011, p. 91), os pacientes terão um registro e cada registro mantém informações de um paciente.

No diagrama de classes de UML também é possível definir multiplicidades, ou seja, definir o número exato de objetos envolvidos no relacionamento, usando

"*". Veja que no diagrama da figura 28, desenvolve esse tipo de diagrama de classe para mostrar que os objetos da classe Paciente também estão envolvidos em relacionamentos com uma série de outras classes. Também é possível nomear as associações para que fique claro ao leitor uma indicação do tipo de relacionamento.

Com a UML é possível, também, especificar o papel dos objetos envolvidos na associação, os diagramas de classe são parecidos com modelos de dados. Segundo Sommerville (2011, p. 91), modelos semânticos de dados são usados no projeto de banco de dados. Eles mostram as entidades dos dados, seus atributos associados e as relações entre essas entidades. Essa abordagem de modelagem foi proposta pela primeira vez em meados da década de 1970, por Chen (1976); desde então, diversas variantes têm sido desenvolvidas todas com a mesma forma básica.

Figura 28. Classes e associações.



Fonte: Sommerville (2011, p.91).

Entretanto, a UML não inclui notações específicas para a modelagem de banco de dados, pois supõe um processo de orientado a objetos e modela dados usando objetos e seus relacionamentos, porém, podemos usar a UML para representar um modelo semântico de dados, como, entidades como classes de objetos, em atributos e em relações como as associações entre classes de objetos.

Quando formos demonstrar as associações entre classes, é importante que seja da maneira mais simples e objetiva possível. Algum detalhe, entretanto, poderá ser adicionado, como informações de atributos e operações:

» O nome da classe de objeto.

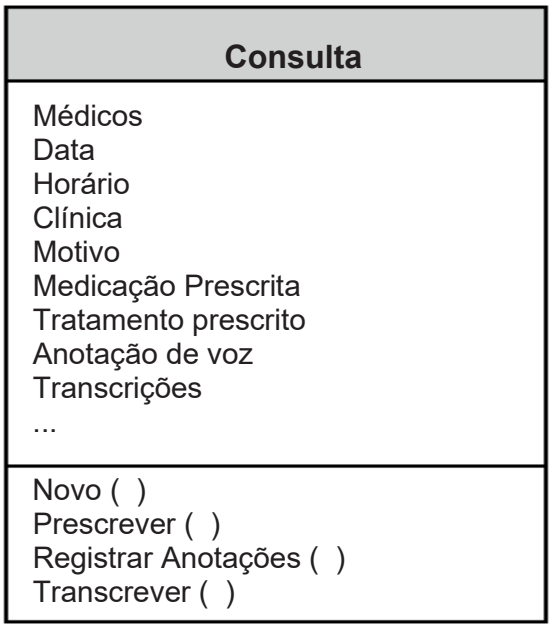
» Os atributos de classe. Pode incluir os tipos de dados.

» As operações ou os métodos (em linguagens de programação).

Por exemplo, um objeto Cliente terá o atributo *Endereço*, e você pode incluir uma operação chamada *Mudar Endereço ( )*, que é chamada quando um cliente indica ter mudado de endereço.

No exemplo sugerido por Summerville (2011, p. 92), a figura a seguir mostra os possíveis atributos e operações da classe Consulta. Nesse exemplo, os médicos registram detalhes da consulta e, para prescrever a medicação, o médico usa o método Prescrever ( ) para gerar uma prescrição eletrônica.

Figura 29. Classe de consulta.

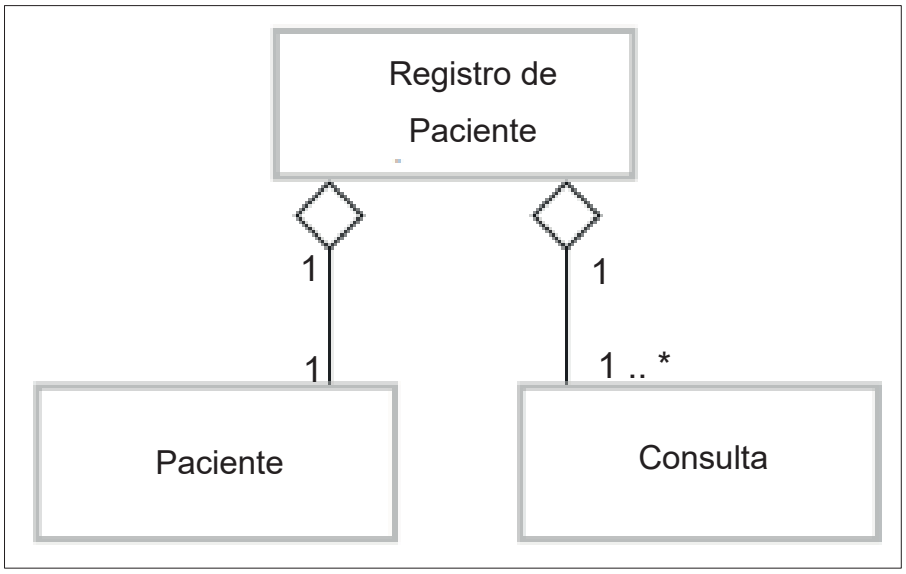


Fonte: Sommerville (2011, p. 91).

Agregação

A UML fornece um tipo especial de associação entre as classes chamada agregação, que significa que um objeto (o todo) é composto de outros objetos (as partes).

Figura 30. A associação por agregação.



Fonte: Sommerville (2017).

A agregação é um relacionamento “parte-todo” ou “uma-parte-de” no qual os objetos que representam os componentes de alguma coisa e que são associados a um objeto que representa a estrutura inteira. A Agregação é desenhada como uma associação exceto por um losango que indica a extremidade estrutural do relacionamento. (RUMBAUGH, 1994, p. 53).

Uma agregação é um subtipo de um relacionamento de associação na UML. Agregação e composição são os dois tipos de relacionamento de associação na UML. Um relacionamento de agregação pode ser descrito em palavras simples como “um objeto de uma classe pode possuir ou acessar os objetos de outra classe”.

Em um relacionamento de agregação, o objeto dependente permanece no escopo de um relacionamento, mesmo quando o objeto de origem é destruído.

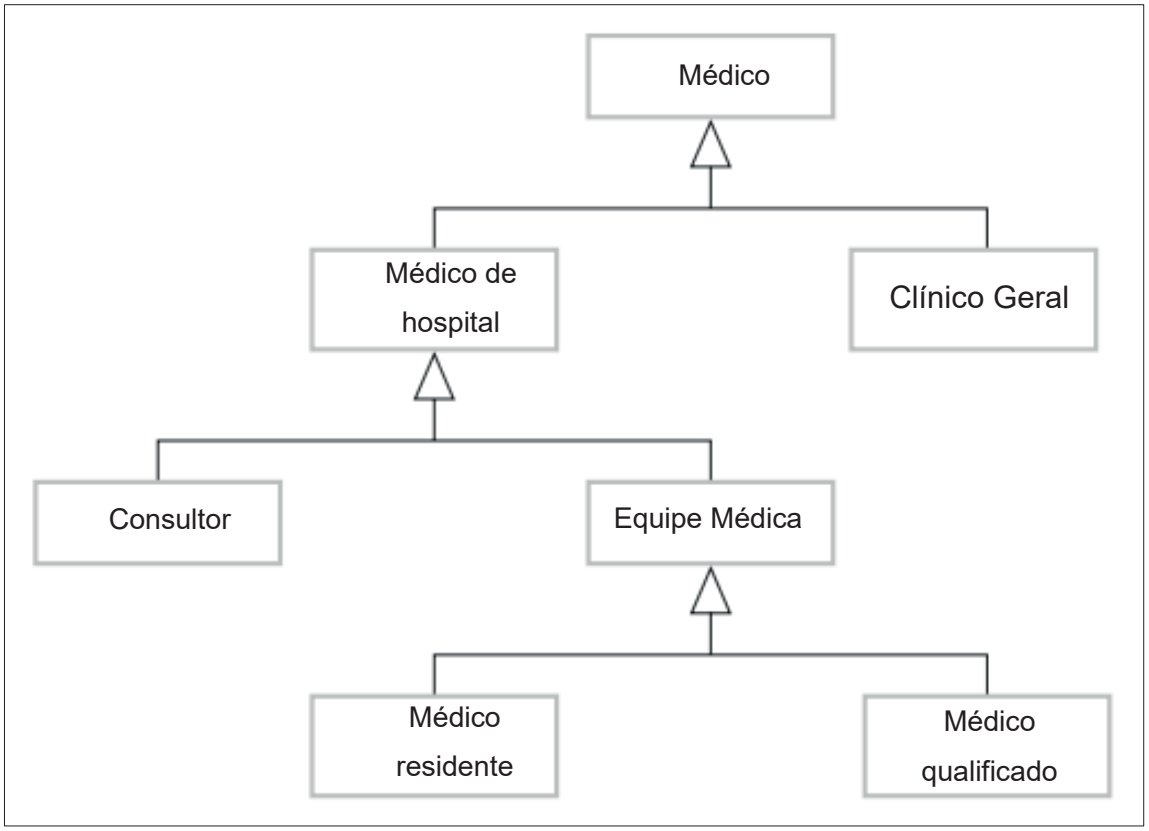
Generalização

Generalização é o processo de extrair características compartilhadas de duas ou mais classes e combiná-las em uma superclasse generalizada. Características compartilhadas podem ser atributos, associações ou métodos.

Segundo Sommerville (2011, p. 92), ao invés de aprender as características detalhadas de cada entidade que nós experimentamos, colocamos essas entidades em classes mais gerais (animais, carros, casas etc.) e aprendemos suas características. Isso nos permite inferir que os diferentes membros dessas classes têm algumas características em comum.

Fernandes (2017) acrescenta que a subclasse herda todas as características, da superclasse, podendo adicionar novos atributos ou operações. Esta relação entre classes é representada por uma seta que parte da subclasse e termina na superclasse:

Figura 31. Hierarquia de generalização.



Fonte: Sommerville (2011, p. 92).

Composição

A composição é um tipo especial de relação de agregação em que as partes componentes não existem, exceto como parte da composição. Em uma Composição, duas entidades são altamente dependentes umas das outras, elas representam parte do relacionamento e, quando há composição entre duas entidades, o objeto não pode existir sem a outra entidade.

Não é um relacionamento UML padrão, mas ainda é usado em vários aplicativos.

- A agregação composta é um subtipo de relação de agregação com características como:
  - é uma associação de mão dupla entre os objetos;

› é um relacionamento todo / parte;

› se um composto for excluído, todas as outras partes associadas a ele serão excluídas.

Referências

FERNANDES, J. M.; MACHADO, R. J. **Requisitos de projetos de software e de sistemas de informação**. Novatec: São Paulo, 2017.

https://www.stickyminds.com/article/requirements-engineering-our-best-practices.

PETERS 2010, James F. **Engenharia de Software**. Rio de Janeiro: Elsevier, 2001.

PFLEEGER, Shari Lawrence. **Engenharia de software**: teoria e prática. 2ª ed. São Paulo: Prentice Hall, 2004.

PRESSMAN, Roger S. **Engenharia de Software**: uma abordagem profissional. 7ª ed. Porto Alegre: AMGH, 2011.

RAMOS, Allan. **Wireframe, Protótipo e Mockup** – é tudo a mesma coisa?. Disponível em: https://medium.com/trainingcenter/wireframe-prot%C3%B3tipo-e-mockup-%C3%A9-tudo-a-mesma-coisa-b990034085d6. Acesso em: 2 setembro 2019.

RUMBAUGH, James *et al*. **Modelagem e Projetos baseados em Objetos**. Rio de janeiro: Campus, 1994.

SOMMERVILLE, Ian. **Engenharia de Software/Ian Sommerville**. Tradução Ivan Bosnic e Kalinka G. de O. Gonçalves. Revisão técnica Kechi Hirama. 9. ed. São Paulo: Pearson Prentice Hall, p. 551, 2011.

STAAB S.; STUDER R. **Handbook on Ontologies**. Springer, 2009.

PMBOK. **Guida do Conhecimento em gerenciamento de projetos**. 6 ed. Project Management Institute, 2017.