ENGENHARIA DE REQUISITOS

Aula 1

Esta aula apresenta conteúdos fundamentais para a formação profissional em engenharia de software, focando na engenharia de requisitos. Ela explora o que é a engenharia de requisitos, por que é classificada como uma "engenharia", seu significado e como se insere na engenharia de software, sendo uma disciplina crucial que pode gerar tanto benefícios quanto prejuízos, afetando inclusive o custo do projeto. Serão abordadas as causas de falhas em projetos de software, como mitigar esses problemas e o papel do analista de requisitos e outros membros da equipe, incluindo suas habilidades e a importância da comunicação.

TEMA 1 – ENGENHARIA DE REQUISITOS

O termo "engenharia" é definido como um conjunto de técnicas e métodos para aplicar conhecimento técnico e científico na planificação, criação e manutenção de estruturas, máquinas e sistemas para benefício humano. A engenharia envolve tarefas contínuas com modelos matemáticos, técnicos e científicos para obter resultados objetivos. Dessa forma, o termo "engenharia de requisitos" é adequado para o processo contínuo de identificar, organizar, documentar e rastrear os requisitos de um software, considerando as necessidades do cliente, que podem mudar. Este processo visa compreender e **explicitar o que é necessário para entregar um software adequado** ao negócio do cliente.

A engenharia de requisitos é a primeira disciplina a ser considerada no desenvolvimento de um novo software, sendo fundamental para todo o trabalho. Os **requisitos são centrais no planejamento do ciclo de vida do projeto**, interagindo com as fases de planejamento, projeto, construção, testes, configuração e manutenção. A avaliação constante dos requisitos em todas as fases garante que as necessidades sejam contempladas na solução. É um processo que engloba todas as atividades para a produção e manutenção de um documento de requisitos, permitindo aos projetistas criar a melhor solução para o cliente.

1.1 Fundamentos de requisitos de software

No mundo atual, softwares são ubíquos, e para serem eficientes e desejados, precisam executar seus objetivos. Para o sucesso de um software e satisfação do cliente, o analista de requisitos deve identificar corretamente as expectativas. Projetos atuais são complexos e envolvem equipes multidisciplinares; a falha em uma parte pode comprometer todo o trabalho.

O guia **SWEBOK (Software Engineering Body of Knowledge)**, patrocinado pela IEEE Computer Society, promove a profissionalização e o consenso sobre a engenharia de software. O **PMI (Project Management Institute)**, uma instituição internacional, aponta a definição incompleta ou malfeita dos objetivos como a maior causa de fracasso de projetos. A qualidade de um software não se mede apenas pela ausência de erros, mas também por atender às expectativas do cliente. Uma boa comunicação com todos os envolvidos é essencial para obter uma lista completa de requisitos.

O desenvolvimento de um sistema exige um plano detalhado de tarefas, organizado ao longo do tempo, para entregar um sistema funcional que atenda às necessidades do cliente, dentro do prazo

e custo. Reuniões frequentes com o cliente e a equipe são essenciais para definir e entender as necessidades. A organização dessas tarefas é o **ciclo de desenvolvimento do projeto, ou ciclo de vida**, com várias abordagens atuais. A documentação de cada fase e o conhecimento dos requisitos são fortemente buscados, independentemente da metodologia. O ciclo de vida apenas organiza as atividades; o analista de sistemas ou líder do projeto gerencia as fases, negocia com usuários e toma decisões.

As **fases do ciclo de vida de um software** incluem: Engenharia de Requisitos, Análise e Projeto, Implementação, Testes, Implantação e Manutenção. A engenharia de requisitos ajuda a organizar essas atividades para produzir um software confiável e de qualidade. A escolha da estratégia de organização depende da complexidade técnica, equipe, cliente e frequência de alterações nos requisitos.

As **estratégias mais conhecidas para organizar um ciclo de vida de projetos** são:

• **Estratégia sequencial (Modelo em Cascata)**: Atividades executadas sequencialmente, uma concluindo antes da próxima. Indicado para projetos com requisitos bem definidos, com foco no planejamento e conferência constante. A desvantagem é o alto custo de correção se um requisito mal especificado for descoberto tardiamente.

• **Estratégia Iterativa/Incremental**: O projeto é desenvolvido em versões completas de partes do software, com entregas frequentes ao cliente. Permite incluir novos requisitos e facilita a correção de falhas sem afetar versões futuras. Os requisitos são organizados por funcionalidades e refinados progressivamente, garantindo maior qualidade. Um exemplo é o **RUP (Rational Unified Process)**, que organiza o desenvolvimento em Iniciação, Elaboração, Construção e Transição. Embora a iniciação tenha grande esforço em levantamento de requisitos, as atividades de requisitos se mantêm até a transição para melhor gerenciamento de mudanças.

• **Metodologia Ágil**: Proposta para desenvolvimento mais rápido e com agilidade na entrega, ideal para projetos menores. O **Manifesto Ágil**, criado em 2001, baseia-se em quatro fundamentos-chave: **indivíduos e interações sobre processos e ferramentas; software funcionando sobre documentação abrangente; colaboração com o cliente sobre negociação de contratos; e resposta às mudanças sobre seguir um plano**. Tem como objetivo entregas rápidas e frequentes, em ciclos chamados **sprints**, onde todo o processo de desenvolvimento (levantamento de requisitos, análise, projeto, implementação, testes e entrega) é realizado para um trecho da solução.

É possível usar mais de uma abordagem em projetos de grande porte, avaliando requisitos, tempo e recursos. As **etapas da engenharia de requisitos** são divididas em dois grupos principais: **desenvolvimento de requisitos** e **gerenciamento de requisitos**. No **desenvolvimento de requisitos** estão:

• **Elicitação**: Técnica para obtenção de dados e identificação de necessidades junto aos **stakeholders** (clientes, empresas, equipes, organizações envolvidas que influenciam os requisitos).

• **Análise**: Avaliação de possíveis conflitos, identificação de relações com o contexto e definição de requisitos.

• **Especificação**: Documentação e detalhamento das especificações dos requisitos.

• **Validação**: Comparação com os propósitos do produto para garantir conformidade e atendimento aos requisitos do cliente. O **gerenciamento de requisitos** visa garantir que todas as necessidades sejam levantadas e devidamente elencadas na documentação.

TEMA 2 – PAPEL DO ANALISTA DE REQUISITOS

O desenvolvimento de sistemas, sejam eles simples ou complexos, é melhor realizado em equipe, onde os integrantes têm papéis definidos por suas especialidades. As funções comuns incluem: analista de negócio, analista de sistemas, analista de requisitos, programadores, testadores e gerente de projeto.

• O **Analista de Negócios** trabalha com a equipe de desenvolvimento, possui conhecimento funcional relevante da área, tem a visão do todo e ajuda a organização a atingir seus objetivos. Ele pode também elicitar e documentar requisitos, assumindo o papel de analista de requisitos.

• O **Analista de Sistemas** possui conhecimento sólido em software, hardware e rede. Suas responsabilidades incluem interagir com usuários finais e clientes, planejar o fluxo do sistema, gerenciar considerações de design e implementação, prazos, e participar ativamente de testes, homologação e implantação, sendo o principal responsável pelo desenvolvimento do projeto e respondendo ao gerente de projetos.

• O **Analista de Requisitos** tem a função de **trazer as necessidades do cliente e de todas as partes interessadas**. É especialista em descobrir essas necessidades, possuindo habilidades de comunicação e técnicas para negociar e interagir com clientes, além de documentar os requisitos para que sejam elicitados.

Outros integrantes da equipe também precisam conhecer os requisitos: os desenvolvedores os usam para codificar e construir funcionalidades, os testadores para gerar planos de testes e validar o atendimento dos requisitos, e o gerente de projeto para gerenciar mudanças e o escopo.

As **responsabilidades e habilidades específicas do analista de requisitos** incluem:

• Adaptar-se e conhecer profundamente o negócio do cliente.

• Levantar informações detalhadas sobre as necessidades do cliente.

• Familiaridade com aplicações e tecnologias relacionadas ao negócio.

• Interagir bem com gerentes, analistas e programadores.

• Boa habilidade analítica, mente desafiadora e curiosa, e atenção aos detalhes.

• Boa escrita e expressão para registrar não conformidades.

• Informar o cliente sobre os requisitos levantados e o andamento do projeto.

Em empresas, um profissional pode acumular as funções de analista de negócios, sistemas e requisitos. No entanto, isso pode gerar sobrecarga e comprometer a qualidade, pois o analista de sistemas, ao acumular o papel de requisitos, pode adiantar a elicitação para não atrasar a próxima fase, o que não é o ideal. É preferível ter profissionais distintos que cooperem e tenham responsabilidades separadas.

TEMA 3 – IMPORTÂNCIA DA ENGENHARIA DE REQUISITOS

Falhas no processo de desenvolvimento de software, causadas por requisitos mal elicitados, podem se manifestar em **atrasos nas entregas, insatisfação do cliente, aumento de custos, retrabalho excessivo, falhas no funcionamento do software, reclamações e desmotivação da equipe**. A citação de Peter Drucker, **"Não há nada tão inútil quanto fazer eficientemente o que não deveria ser feito"**, ilustra a importância de se ter requisitos claros.

Um cenário comum é uma equipe trabalhando intensamente para entregar um software, mas o cliente reclama que o produto final não era exatamente o que ele queria, faltando uma funcionalidade essencial. Isso pode levar a grandes retrabalhos, mudanças estruturais que inviabilizam a manutenção ou até a necessidade de reconstruir quase tudo, gerando **prejuízos financeiros e marketing negativo irreversível**.

Saber exatamente o que o cliente quer, precisa e espera é fundamental para um resultado eficiente e lucrativo. A engenharia de requisitos provê informações cruciais em todas as fases do projeto, independentemente da estratégia. Mesmo com equipes bem treinadas e tecnicamente especializadas, o problema pode ser a **produção "da coisa errada"** se a devida atenção não for dada à engenharia de requisitos, que **direciona a equipe para atender às necessidades do cliente**.

TEMA 4 – IMPACTOS NEGATIVOS DA FALHA EM REQUISITOS

Ainda ocorrem retrabalhos, atrasos e insatisfação do cliente devido a falhas nos requisitos. O **Project Management Institute (PMI) constatou que 47% dos projetos fracassados são causados por deficiência na engenharia de requisitos**, e esse número pode ser ainda pior para projetos de software, onde o retrabalho causado por falhas de requisitos pode não ser visível.

Quanto mais tarde os requisitos incompletos ou errados são descobertos, maior o custo para corrigir o projeto. Estudos de Barry Boehm indicam que o custo para corrigir um defeito **após a entrega pode ser 100 vezes maior** do que corrigi-lo no início, durante a elaboração dos requisitos. Outros estudos mostram que defeitos originados em requisitos representam uma porcentagem significativa do total de erros (Capers Jones: 20%; Pohl: até 60% em projetos complexos), e esses erros são frequentemente detectados em fases avançadas do trabalho, como na homologação.

A **documentação de requisitos é vital** para reter o conhecimento do negócio da empresa, minimizar perdas de informação e prejuízos quando as equipes mudam. Estudos da Escola de Gerenciamento de Cranfield mostram que 68% dos projetos são destinados ao insucesso.

Algumas **causas comuns para o insucesso de projetos** incluem:

• **Planejamento inadequado**: Desvios não previstos no planejamento detalhado de metas, cronograma, tarefas e recursos.

• **Definição do projeto inadequada**: Requisitos errados ou incompletos comprometem todo o projeto.

• **Definição do escopo inadequada**: Alterações nos itens essenciais podem atrasar, inviabilizar e aumentar custos.

• **Alteração das especificações do projeto**: Mudanças após o início das atividades impactam diretamente nas tarefas futuras e no que já foi produzido, gerando custos e atrasos.

• **Inexperiência dos gestores do projeto**: Incompetência no gerenciamento de custos e recursos pode comprometer a conclusão e desmotivar a equipe.

• **Cronograma e expectativas irreais**: Prazos apertados causam estresse, impactam a qualidade e geram falhas graves.

• **Falta de suporte e envolvimento da alta gestão**: A ausência de patrocínio gerencial prejudica a tomada de decisões rápidas e assertivas.

• **Falta de envolvimento do cliente**: A não participação do cliente em todo o desenvolvimento pode levar à insatisfação e ao fracasso do produto final.

Problemas com requisitos incompletos impactam a aceitação do software pelo cliente, podendo levar ao cancelamento do serviço ou do projeto, e resultam em **baixa qualidade do produto**, com usuários sentindo falta de funcionalidades ou funcionamento diferente do esperado.

TEMA 5 – DIFICULDADES COMUNS COM REQUISITOS

A coleta de requisitos, embora pareça precisa, enfrenta diversas dificuldades práticas:

• **Requisitos nem sempre são óbvios** e podem vir de várias fontes.

• **Dificuldade dos clientes em expressar requisitos claramente**: Muitos clientes não sabem explicar suas necessidades ou não as identificam como requisitos importantes.

• **Diversidade de tipos de requisitos** em diferentes níveis de detalhe.

• **Crescimento desordenado do número de requisitos**, dificultando o gerenciamento.

• **Várias partes interessadas (stakeholders)**, exigindo gerenciamento por grupos com diferentes funções.

• **Requisitos são alterados** ao longo do desenvolvimento, mudando ou aumentando o escopo inicial.

A capacidade de expressão do cliente é uma das maiores dificuldades. O analista de requisitos tem o papel de compreender corretamente as necessidades, mesmo que o cliente não consiga comunicá-las adequadamente. Os **maiores desafios no desenvolvimento de software, em relação aos requisitos, estão ligados à comunicação, e não a questões técnicas**.

É crucial que o **analista de requisitos aprenda sobre o negócio do cliente antes mesmo de interagir com ele**. Isso permite formular perguntas mais assertivas, identificar erros ou informações incompletas, e perceber o que é óbvio para o cliente, mas não foi dito. As **partes interessadas (stakeholders)**, pessoas ou organizações envolvidas no projeto ou afetadas por ele, devem ser ouvidas. A não identificação de um stakeholder pode aumentar substancialmente os custos (por exemplo, requisitos legais do departamento jurídico). Sem essas informações, os requisitos podem ser incompletos, resultando em uma solução final inútil ou incompatível com as partes interessadas.

Mudanças sempre ocorrem, e o analista de requisitos deve estar atento para incorporá-las ou avaliá-las o quanto antes, evitando problemas futuros. Apesar das dificuldades, o **analista de requisitos deve estar preparado com ferramentas, técnicas, habilidades e bom**

relacionamento interpessoal para elicitar todas as necessidades do produto de software, que é sua principal meta.

Aula 2

Esta aula aborda os tipos e características dos requisitos, detalhando técnicas para sua obtenção junto a representantes do cliente, usuários, equipe do projeto e outras partes interessadas, ou *stakeholders*. Ela define os *stakeholders*, sua importância no levantamento de requisitos e a classificação dos requisitos em diferentes categorias. Também serão consideradas as restrições e premissas que servem de base para um projeto, e a diferença entre esses fundamentos para o conceito de requisito. A aula enfatiza que requisitos de alta qualidade devem ser claros, completos, sem ambiguidade, consistentes e testáveis, destacando a importância de uma boa elicitação e especificação documental. Por fim, são avaliadas técnicas para levantar, detalhar, documentar e validar requisitos, sempre focando no escopo definido, que estabelece as fronteiras do que será e não será implementado. Uma boa engenharia de requisitos é fundamental para um projeto de software de qualidade.

TEMA 1 – DEFINIÇÃO DO ESCOPO: DOMÍNIO DO PROBLEMA E REQUISITOS DE NEGÓCIO

O termo "escopo" refere-se ao ponto de mira, alvo, objetivo ou âmbito de um projeto. Segundo o guia PMBOK, a declaração do escopo do projeto é um documento final que detalha o que será entregue, prazos, custos e atividades, promovendo um entendimento comum entre o cliente e a equipe de desenvolvimento. O escopo deve garantir que todo o trabalho necessário, e apenas o necessário, seja contemplado para o sucesso do projeto, explicitando o que será feito e o que não será. O **escopo positivo** descreve tudo o que o projeto deve atender e entregar como produto final, incluindo produtos, serviços e resultados esperados. O **escopo negativo** identifica claramente o que o projeto não irá atender ou entregar, sendo crucial para gerenciar expectativas e evitar serviços não citados no contrato.

O **domínio do problema** é a área em análise que corresponde às fronteiras (decisões, políticas) em que o software estará imerso e com as quais precisará interagir. Ele engloba as metas da organização e as razões para iniciar o projeto, bem como os problemas a serem resolvidos.

Uma boa declaração de escopo deve:

• Descrever o software a ser desenvolvido e as características do produto.

• Definir os critérios de aceitação e como as verificações serão feitas.

• Estabelecer o cronograma de entregas, detalhando entregas parciais e documentação.

• **Descrever o escopo negativo** para gerenciar expectativas.

• Descrever as restrições e premissas do projeto, incluindo riscos potenciais.

Quanto mais completa a definição do escopo, menos problemas e conflitos tendem a ocorrer entre as partes interessadas e a equipe de desenvolvimento.

TEMA 2 – RESTRIÇÕES E PREMISSAS

Este tema aborda fatores que podem influenciar ou limitar o desenvolvimento do projeto.

2.1 Restrições

Restrições são limitações que afetam as possíveis soluções de desenvolvimento do software e impactam a implementação, testes, validação e implantação. Elas não são negociáveis e não podem ser modificadas, apenas identificadas, tratadas e validadas. Um exemplo é um prazo legal para a entrega do software, que pode levar a uma solução mais rápida, mas não a ideal. Além de restrições de negócio, existem restrições tecnológicas, como decisões sobre arquitetura, uso de servidores, memória e processadores, que são essenciais para evitar soluções inadequadas.

2.2 Premissas

Premissas são suposições que, para fins de planejamento, são consideradas verdadeiras, reais e certas sem necessidade de prova. Elas devem ser validadas e confirmadas para prosseguir com a especificação de requisitos. Um exemplo é o uso do CPF como chave de login, presumindo que todos os usuários terão um CPF único. Se essa premissa for falsa (ex: menores de idade sem CPF), pode exigir retrabalho significativo.

2.3 Riscos

Premissas podem configurar **riscos** ao projeto. Riscos são eventos incertos que, se ocorrerem, podem causar efeitos negativos. É fundamental identificá-los para planejar ações preventivas e minimizar problemas. A avaliação de riscos determina a probabilidade de ocorrência e o grau de impacto, permitindo à equipe planejar reações. A Figura 4 (no material de origem) ilustra a diferença entre premissas (ex: "Hoje não vai chover"), restrições (ex: "Fazer a festa em local coberto") e riscos (ex: "Chover durante a festa").

TEMA 3 – PARTES INTERESSADAS (STAKEHOLDERS)

O termo **stakeholders** (partes interessadas) surgiu para classificar aqueles que podem influenciar decisões em uma organização. Na Engenharia de Requisitos, foi adotado para determinar todos que influenciam os requisitos de um sistema e são afetados por ele, indo além de apenas cliente e usuário. O nível de interesse e o grau de influência (poder) dos *stakeholders* são atributos que devem ser considerados e gerenciados.

3.1 Cliente vs. usuário

Cliente é quem solicitou e paga pelo serviço, esperando que suas necessidades sejam atendidas no prazo e custo. **Usuário** é quem utilizará o produto desenvolvido para realizar o trabalho da organização, esperando um produto útil, fácil de usar e eficiente. No entanto, muitos requisitos vêm de outras partes interessadas que não são clientes ou usuários. Exemplos de *stakeholders* incluem acionistas, colaboradores, gestores, concorrentes, fornecedores, governos e a mídia. O analista de requisitos precisa interagir com todos os *stakeholders*, mesmo aqueles que podem não estar interessados na solução ou podem enxergar o sucesso do projeto como algo negativo (os "stakeholders negativos"). Negligenciar esses *stakeholders* pode aumentar a probabilidade de falha do projeto. O documento de especificação de requisitos deve ser claro, objetivo e sem excesso de

termos técnicos, para que o cliente o entenda e possa avaliar se suas necessidades foram contempladas.

TEMA 4 – REQUISITOS

4.1 Definição de requisitos

Na fase inicial de um projeto, o foco é compreender o que o software deve fazer para satisfazer as necessidades de quem o solicitou. O objetivo é buscar o máximo de informações e entender a natureza do software, o domínio do problema e o comportamento esperado do sistema. O **IEEE (Instituto de Engenheiros Eletricistas e Eletrônicos)** define requisito como uma condição ou capacidade do sistema solicitada por um usuário para resolver um problema, ou uma condição/capacidade que deve ser atendida por uma solução para satisfazer um contrato, especificação, padrão, ou outros documentos formais. Inclui as necessidades, desejos e expectativas documentadas do patrocinador, clientes e *stakeholders*. A definição dos requisitos deve ser suficientemente abstrata para não direcionar prematuramente para uma solução específica ou tecnologia, focando nas reais necessidades do cliente e no comportamento esperado do software.

4.2 Tipos de requisitos

Existem diversas classificações de requisitos organizadas hierarquicamente.

4.2.1 Requisitos de negócio

Os **requisitos de negócio**, também conhecidos como regras de negócio, definem a forma como o negócio do cliente opera. Eles refletem a política interna da empresa, processos existentes e normativas que os funcionários já seguem, e que o software deve contemplar. Uma regra de negócio não é necessariamente uma funcionalidade do software, mas guia o comportamento das funcionalidades. Nem todas as regras de negócio são automatizadas por um sistema. Exemplos incluem critérios para conceder limite de cheque especial ou aceitação de tipos de pagamento.

4.2.2 Requisitos das partes interessadas

São as necessidades e expectativas dos *stakeholders* em relação à organização. A avaliação desses requisitos ajuda a identificar conflitos e consolidar necessidades semelhantes. Exemplos incluem as expectativas dos pais dos noivos em um casamento, dos convidados sobre a festa, ou de fornecedores sobre pagamentos. Para uma empresa, podem ser as expectativas dos funcionários sobre salários ou dos órgãos reguladores sobre o cumprimento da legislação. É crucial definir os requisitos pertinentes ao sistema de gestão da qualidade da organização, identificando quem afeta esse sistema.

4.2.3 Requisitos de transição

São requisitos temporários, importantes para a implantação da solução, mas necessários somente para que a transição seja possível. Eles são previstos após o projeto do novo software e deixam de ser relevantes quando o sistema antigo é desativado. Envolvem tarefas como conversão de dados, treinamentos de usuários e capacidades como redundâncias ou trabalhos paralelos. São comparados a andaimes de uma obra, que são descartados após o uso. Exemplos incluem recrutamentos, mudanças de setor, migração de dados e treinamentos.

4.2.4 Requisitos da solução

Descrevem as características da solução que atendem aos requisitos de negócio e das partes interessadas, capturando decisões sobre o escopo e o comportamento esperado do software. Subdividem-se em funcionais e não funcionais.

4.2.4.1 Requisitos funcionais

Definem uma função do sistema de software ou de um componente, representando o que o software faz em termos de tarefas e serviços. Descrevem o comportamento, tarefas e serviços que o usuário terá para interagir com o software, incluindo o processo de comunicação, armazenamento e o objetivo a ser alcançado. Exemplos incluem "O sistema deve manter o cadastro de alunos" ou "O sistema deve possibilitar o registro de notas por disciplina".

4.2.4.2 Requisitos não funcionais

Descrevem limitações aos requisitos funcionais, abordando como as funcionalidades serão entregues ao usuário. Também conhecidos como requisitos de qualidade ou suplementares, eles descrevem condições como desempenho, usabilidade, confiabilidade, segurança, disponibilidade, manutenção e tecnologias envolvidas. Um requisito não funcional não atendido pode inutilizar um software, como um tempo de resposta excessivo de uma pesquisa. Podem surgir de necessidades de usuários, orçamento, políticas organizacionais ou fatores externos. Exemplos incluem tempo máximo para resposta (desempenho), disponibilidade 24/7 (disponibilidade), privilégios de acesso (segurança), e tempo de implementação de modificações (manutenibilidade). Além desses, existem os **requisitos derivados**, que fazem parte do modelo de solução adotado e são importantes para a etapa de codificação/implementação, decorrendo de requisitos técnicos. Por exemplo, "O sistema deve funcionar 24 horas, no Alasca" implica requisitos derivados como "O sistema deve funcionar na neve" e "O sistema deve ter baterias para nunca ficar sem energia elétrica".

TEMA 5 – DOCUMENTO DE ESPECIFICAÇÃO DE REQUISITOS

O **documento de especificação de requisitos** é fundamental, servindo como um contrato entre o cliente e a equipe de desenvolvimento. Ele registra os requisitos selecionados, restrições e regras de negócio que o projeto deve atender. É crucial descrever os requisitos em vários níveis de detalhe para que todas as partes interessadas (cliente, programador, etc.) possam entendê-los a partir de sua própria perspectiva.

O analista de requisitos tem o desafio de compreender as diversas perspectivas e necessidades dos *stakeholders*, identificar omissões, inconsistências ou falhas de entendimento nos processos, para evitar retrabalho e perda de recursos. Na equipe de desenvolvimento, o documento é usado pelo gerente de projetos (para planejamento, cronograma, custos), pelo analista projetista (para arquitetura e recursos tecnológicos) e pelos analistas de teste (para planos de teste e validação do código). Erros na especificação impactam todas essas funções e podem gerar prejuízos.

Para o sucesso, o analista de requisitos deve se preparar bem, identificar todos os *stakeholders*, desenvolver competências interpessoais (mediação de conflitos, persuasão, empatia, comunicação clara) e ter alta capacidade de escuta e compreensão. Padrões para a escrita do documento incluem:

• Iniciar com "O sistema deve..."

• Usar frases curtas.

• Atribuir um identificador único a cada requisito.

• Separar requisitos funcionais dos não funcionais.

• Evitar detalhes de implementação.

• Manter a consistência na terminologia do domínio.

Recomenda-se elaborar dois documentos distintos: um para o cliente, em linguagem natural, e outro para a equipe de desenvolvimento, em linguagem mais técnica. A correspondência entre a definição e a especificação dos requisitos é vital para a rastreabilidade, possibilitando o planejamento de casos de teste e a gestão de mudanças. A Figura 15 (no material de origem) apresenta um exemplo de tabela de requisitos funcionais. O documento deve ser completo e coerente, servindo como comprovação de tudo o que foi acordado para o projeto.

Em resumo, despender tempo e realizar uma boa análise de requisitos é crucial para garantir a satisfação do cliente e minimizar problemas futuros no software

Aula 3

Esta aula, ministrada pela Profª Rosemari Pavan Rattmann, aprofunda-se na forma como a descrição dos requisitos deve ser realizada, considerando o nível de detalhamento e as principais técnicas para sua **elicitação e comunicação eficiente**. A professora enfatiza que requisitos devem ser claros, detalhados e simples, mas que, na prática, o analista de requisitos enfrenta **divergências e opiniões diferentes** entre os *stakeholders*, e a linguagem natural nem sempre é adequada para explicitá-los. O objetivo é compreender a elicitação de requisitos, o trabalho dos *stakeholders*, e as técnicas de comunicação para capturar corretamente todas as necessidades pertinentes.

TEMA 1 – ESPECIFICAÇÃO

A **especificação de requisitos** é o processo de escrever os requisitos do usuário e do sistema em um documento. Embora os requisitos devam ser claros, detalhados e compreensíveis por todos, na prática, os *stakeholders* frequentemente apresentam opiniões e necessidades divergentes, resultando em **conflitos de interesse, financeiros e pessoais**. Cabe ao analista de requisitos desenvolver técnicas de comunicação para garantir o melhor resultado na especificação.

A especificação documenta diferentes tipos de requisitos, valida as necessidades das partes interessadas, auxilia na definição da solução de negócio, apresenta o entendimento da equipe do projeto, **garante a qualidade dos requisitos e viabiliza a auditoria das funcionalidades**. Existem diferentes momentos para redigir documentos de especificação, conforme as necessidades do projeto.

Para explicitar os requisitos, utilizam-se notações como **linguagem natural, diagramas e tabelas**. Requisitos de sistema e regras de negócio podem demandar outras formas de escrita, como cálculos, gráficos e modelos matemáticos. A fonte detalha e exemplifica quatro tipos de notações para escrever requisitos:

• **Sentenças em linguagem natural**: Frases numeradas, cada uma expressando um requisito. Exemplos incluem "O sistema deve permitir limpar os dados que estarão digitados na tela".

• **Linguagem natural estruturada**: Requisitos escritos em linguagem natural dentro de um *template*, onde cada campo fornece informações sobre um aspecto do requisito. Um exemplo detalha uma função, descrição, entrada, saída, ação e pré-condição.

• **Notações gráficas**: Modelos gráficos, complementados por anotações de texto, usados para definir requisitos funcionais. **Diagramas de casos de uso e de sequência da UML** são frequentemente empregados. A fonte apresenta um exemplo de diagrama de caso de uso [12, Figura 4].

• **Especificações matemáticas**: Baseadas em conceitos matemáticos (máquinas de estado finitos, conjuntos), embora de difícil compreensão para os clientes. Um exemplo é a Tabela Trauma Score Modificado [12, Figura 5].

Os **requisitos de usuário** devem ser compreensíveis tanto para usuários quanto para a equipe de desenvolvimento. Já os **requisitos de sistema** são versões mais amplas e detalhadas dos requisitos de usuário, fornecendo informações suficientes para que os engenheiros de software projetem a solução. O documento de especificação de requisitos pode ser usado como um contrato e, portanto, deve ser completo, detalhando o sistema inteiro.

O **Documento de Requisitos de Software** é uma declaração oficial de tudo que os desenvolvedores devem implementar, com detalhamento suficiente, sendo imprescindível para contratações externas ou sistemas complexos. Métodos ágeis, por sua vez, podem não abordar o documento de requisitos de forma tão detalhada devido à natureza mutável dos requisitos, priorizando usuários e entregas incrementais. No entanto, um **documento de suporte** que descreva os requisitos e o comportamento esperado do sistema é sempre útil, devendo ser avaliado com as versões implementadas.

TEMA 2 – NÍVEL DE DETALHAMENTO DA ESPECIFICAÇÃO

O nível de detalhamento dos requisitos deve ser ajustado com base em fatores como **porte do sistema, quantidade de *stakeholders*, experiência da equipe e padrões da empresa**. Um detalhamento insuficiente pode levar a interpretações errôneas e falta de funcionalidade, enquanto um **exagero pode direcionar a equipe para soluções inadequadas**. O desafio é encontrar o equilíbrio.

O nível de aprofundamento é determinado pelos objetivos do projeto. Inicialmente, o conhecimento sobre as necessidades é limitado, e decisões sobre o escopo podem ser alteradas. Exemplos mostram uma **evolução no detalhamento**: de um requisito inicial vago como "criar um cadastro de pacientes", para a necessidade de identificar pacientes individualmente e registrar procedimentos, até a inclusão de requisitos de sigilo e limites de idade para informações de prontuário, ampliando o escopo com maior nível de detalhe. A especificação pode variar de uma descrição geral do escopo a um detalhamento completo de interações, armazenamento de dados e regras. O analista de requisitos, ao longo do tempo, descobre e anota novos detalhes que completam a lista de requisitos.

A equipe de desenvolvimento espera um detalhamento da arquitetura e implementação para otimizar o tempo e a codificação. Contudo, incluir muitos detalhes de implementação na fase de

especificação de requisitos é um erro, pois podem ser descritos em outras fases do projeto. **Detalhar excessivamente pode "engessar" a solução**, desperdiçando tempo e recursos. A ausência de informações, por outro lado, gera uma solução incompleta.

A fonte apresenta fatores que influenciam o nível de detalhe adequado [25, Figura 6]:

• **Menos Detalhamento**: Desenvolvimento interno, equipe agrupada, sem casos de testes elaborados nesta fase, estimativas menos precisas, baixa rastreabilidade, alto envolvimento dos clientes, alto conhecimento da equipe sobre o negócio, baixa expectativa de rotatividade de recursos, novos processos operacionais.

• **Mais Detalhamento**: Desenvolvimento externo, equipe dispersa, casos de testes elaborados em paralelo, estimativas mais precisas, alta rastreabilidade, baixo envolvimento dos clientes, baixo conhecimento da equipe sobre o negócio, alta expectativa de rotatividade de recursos, processos operacionais definidos e maduros.

TEMA 3 – CRITÉRIOS DE QUALIDADE DA ESPECIFICAÇÃO

Não existe uma especificação perfeita e completa, e os recursos são finitos. O analista de requisitos deve focar em elaborar uma especificação **boa e compreensível** para todos os envolvidos, ajudando a equipe de desenvolvimento a entender o que os usuários realmente querem.

A norma **IEEE 830** (citada por Vasquez e Simões, 2016) fornece os critérios de qualidade para uma boa especificação:

• **Correta**: Cada requisito deve satisfazer uma necessidade do negócio, com **rastreabilidade**. Requisitos sem relação com o negócio devem ser eliminados.

• **Completa**: Todos os elementos significativos do domínio do problema devem ser descritos, sem deixar partes "a definir".

• **Clara**: Não deve haver ambiguidades ou múltiplas interpretações. A linguagem natural pode ser ambígua, sendo útil um **glossário de termos**.

• **Consistente**: A especificação não deve conter contradições entre os requisitos. Isso é comum com múltiplos analistas ou alterações mal gerenciadas.

• **Modificável**: As alterações devem ser fáceis, completas e consistentes, sem comprometer a estrutura. A **rastreabilidade dos requisitos** e um glossário são úteis.

• **Priorizada**: Cada requisito recebe um valor de prioridade baseado em relevância para a solução, *stakeholders*, custos, riscos e prazos. O objetivo é focar nos **requisitos mais críticos** e reduzir riscos.

• **Verificável (Testável)**: Deve ser possível usar um método para demonstrar que a solução satisfaz o requisito. Um requisito como "o sistema deve possibilitar a utilização de todos os navegadores" não é verificável eficientemente.

• **Rastreável**: Ajuda a validar requisitos em relação à solução, descobrir falhas, requisitos ausentes ou incorretos, e garantir que todos os objetivos de negócio sejam atendidos para todas as partes interessadas.

Estes critérios auxiliam o analista na elaboração e serão aplicados principalmente nas atividades de verificação e validação de requisitos.

TEMA 4 – ELICITAÇÃO

Elicitação de requisitos é o termo mais atual para o levantamento de requisitos, significando a **ação ou efeito de elicitar, de obter informações detalhadas sobre o que se pretende fazer**. É um processo de aquisição de conhecimento que utiliza técnicas para buscar, entender, obter, descobrir e elaborar requisitos, identificando fatos para um **completo entendimento do que é esperado** pelas partes interessadas. Envolve comunicação, priorização, negociação e colaboração com todos os *stakeholders*.

As atividades comuns de elicitação incluem: compreensão do domínio da aplicação e identificação das fontes de requisitos; análise das partes interessadas; seleção de técnicas, abordagens e ferramentas; e elicitação dos requisitos. Este processo é **iterativo**, ou seja, é refeito e repetido, exigindo avaliação contínua para refinar as informações e compreender o negócio da organização.

4.1 Dificuldades da elicitação

A organização e agrupamento das informações, separando *stakeholders* em grupos para entender diferentes pontos de vista, minimiza as dificuldades. Um **ponto de vista** organiza um conjunto de requisitos de um grupo de *stakeholders* com algo em comum. As opiniões e prioridades dos *stakeholders* podem ser divergentes. O analista de requisitos deve organizar reuniões para mediar acordos e garantir que todos se sintam atendidos. A documentação deve ser simples e objetiva para ser compreendida por todos.

Ao final da elicitação, a documentação ainda não é padronizada e precisa passar por uma **fase de análise de requisitos** para adequar, reunir, padronizar e entregar à equipe de desenvolvimento. A análise de requisitos avalia a qualidade, completude e clareza dos requisitos antes de repassá-los.

4.2 Preparação para elicitação

A necessidade de documentar requisitos em diferentes momentos exige **múltiplas conversas e reuniões** com *stakeholders*. O analista de requisitos precisa de habilidades de **persuasão e mediação de conflitos**. Planejar as reuniões, definindo participantes, assuntos, objetivos e perguntas, além de conhecer o público antecipadamente, é crucial.

4.3 Execução da elicitação

Nesta fase, as informações são levantadas utilizando técnicas pré-selecionadas, **mantendo o foco no escopo do projeto**. O analista deve registrar atributos dos requisitos (origem, prioridade, autor, proprietário, risco) e documentar tudo para revisão e validação. É fundamental descobrir requisitos implícitos e antecipar desejos dos *stakeholders* para evitar retrabalhos futuros.

4.4 Documentação dos resultados da elicitação

O objetivo é registrar os resultados, anotar informações, confirmar o entendimento do negócio e o que foi relatado pelos *stakeholders*. É preciso estabelecer uma lista de requisitos a serem confirmados e validados, registrar questões não respondidas e dúvidas. Os registros das decisões tomadas devem ser organizados por assunto.

4.5 Confirmação dos resultados da elicitação

Esta fase garante que cada requisito identificado foi bem compreendido pelos *stakeholders*, superando problemas de comunicação. A fonte menciona um "protocolo do garçom" como exemplo de protocolo de comunicação para receber, documentar e confirmar informações [51, Figura 9].

TEMA 5 – TÉCNICAS DE COMUNICAÇÃO

Os problemas de elicitação de requisitos dependem do **contexto social e das pessoas**, não apenas da tecnologia. O analista de requisitos precisa **compreender o trabalho dos *stakeholders***, o domínio da aplicação, atividades, serviços, características e limitações para garantir que o software seja útil e eficiente.

As dificuldades nessa fase são muitas: *stakeholders* nem sempre têm uma ideia clara do que precisam ou dificuldade em descrever seu conhecimento; o analista pode ter pontos de vista diferentes; e usuários podem não ter tempo ou disposição para participar efetivamente. Sommerville (2018) lista outras razões para a dificuldade: *stakeholders* não saberem o que esperar ou proporem exigências inviáveis; requisitos expressos em termos não compreensíveis; influência de fatores políticos; mudanças em variáveis ambientais/de negócio; e surgimento/mudança de prioridade de requisitos.

Para atingir os objetivos, o analista pode usar várias **técnicas de comunicação**, classificadas em tradicionais, colaborativas, cognitivas e de abordagens contextuais [55, 56, Figura 10].

5.1 Técnicas tradicionais

• **Análise de documentação existente**: Estudo de documentos da empresa (financeiros, relatórios, pesquisas, marketing, ouvidoria) para entender processos pertinentes ao escopo.

• **Entrevista**: Preparar questões para *stakeholders* e conduzir conversas detalhadas. O desafio é criar um ambiente de confiança. Embora importante para um entendimento geral, não é eficaz para elicitar conhecimentos específicos de requisitos e geralmente é insuficiente, necessitando de outras técnicas complementares. Vantagens incluem riqueza de informações, investigação aprofundada e contato direto; desvantagens são dificuldade de analisar dados qualitativos e comparar respondentes, e a complexidade da habilidade de entrevistar.

• **Questionários**: Úteis para preparar entrevistas, complementar informações e lidar com grande número de envolvidos de forma prática e rápida, sem interação direta.

• **Histórias e Cenários / Casos de Uso**: Pessoas têm dificuldade com descrições abstratas, mas são capazes de descrever suas atividades e imaginar novas formas de trabalho.

  ◦ **História**: Descrição narrativa das atividades das partes interessadas.

  ◦ **Cenário**: Formato mais estruturado com informações específicas, desenhos e imagens, descrevendo como o usuário interage com o sistema. Um cenário deve incluir: estado do sistema antes, fluxo normal de eventos, exceções, atividades concorrentes e estado final do sistema.

  ◦ **Casos de Uso**: Cenários diagramados de forma mais abstrata, identificando atores, funcionalidades principais e interações entre eles [64, Figura 12].

5.2 Técnicas colaborativas

• **Brainstorming (Tempestade de Ideias)**: Dinâmica de grupo onde os participantes lançam ideias sobre um assunto para chegar a uma conclusão. O analista anota, provoca discussões, media conflitos e estabelece limites. Utiliza estímulos visuais como filmes, quadros, *mockups* e protótipos [65, Figura 12].

• **JAD/RAD (Joint/Rapid Application Development)**: *Workshops* intensivos (dias a semanas) onde usuários e analistas se reúnem para documentar requisitos. Utiliza meios visuais e a abordagem WYSIWYG ("O que você vê é o que você obtém"), gerando documentos de fácil compreensão e acordados.

• **Prototipação**: Criação de uma versão inicial (*draft*) do sistema para experimentação pelos usuários, permitindo uma visão clara do resultado final, verificação de falhas e informações faltantes, e avaliação de viabilidade e custos. É útil para usuários com dificuldade em visualizar a solução.

◦ **Prototipação em papel**: Utiliza meios físicos para demonstrar o funcionamento e as interfaces, permitindo sugestões e alterações do usuário [69, Figura 13].

◦ **Prototipação executável**: Desenvolve um protótipo funcional usando linguagens de quarta geração ou ambientes de prototipagem rápida, simulando o funcionamento. Exemplos incluem ferramentas como Axure, que geram protótipos navegáveis para teste de funcionalidades [70, Figura 14].

5.3 Abordagens cognitivas

• **Análise de tarefas**: Acompanha a rotina de trabalho dos usuários para compreender atividades complexas ou mal explicadas.

• **Análise de protocolos**: Participantes realizam uma tarefa cotidiana enquanto descrevem o que estão fazendo e seu processo de pensamento, revelando problemas de interação em sistemas existentes.

5.4 Abordagens contextuais

• **Etnografia**: Técnica das ciências sociais que se mostra útil para entender os **processos reais de trabalho** através da observação, pois as pessoas geralmente acham difícil descrever o que fazem, e a prática real pode diferir dos processos formais [72, Figura 15].

Finalizando, esta aula aborda a importância do detalhamento dos requisitos, os critérios de qualidade para uma boa especificação, e diversas técnicas para elicitar e comunicar requisitos, reconhecendo que os problemas de elicitação dependem fortemente do contexto social e humano, não apenas da tecnologia