

OUTRAS AULAS EM:

www.projetoderedes.com.br

Sistemas de Informação

Disciplina: Engenharia de Software e da Usabilidade

Professor: José Maurício S. Pinheiro

AULA 3 – Engenharia de Requisitos

Muitos motivos influenciam o início de um projeto de desenvolvimento de um sistema de informação. Em geral, podemos dizer que um projeto é iniciado quando o benefício do retorno esperado supera o custo do projeto.

Também é importante entender que sistemas de informação são interativos e reativos. Interativo significa que o sistema troca informações com o ambiente, em especial com os agentes externos que fazem parte desse ambiente, pessoas e outros sistemas computacionais. O sistema só faz sentido se é capaz dessa interação. Reativo significa que o sistema funciona reagindo a mudanças no ambiente, em especial, a mudanças provocadas pelos agentes externos.

Também são sistemas de respostas planejadas. Isso significa que nossas respostas são determinadas e determinísticas, que podemos criar um programa que as produza. Também significa que todas as perguntas que podem ser feitas ao sistema devem ser identificadas previamente.

Escolhendo essas regras de modelagem determinamos que o sistema só funciona para responder um evento no ambiente, causado por um agente externo, e que possui uma resposta planejada.

1. Sistemas de Informação e a Organização

Sistemas de informação operam em todas as áreas e níveis das organizações, sendo considerados como ferramentas essenciais para o sucesso de suas atividades. Isso nos permite classificá-los de acordo com a responsabilidade assumida por seus usuários dentro da organização em quatro tipos principais, como sugerido por Laudon (Fig. 1):

- **Sistemas de nível operacional** - tratam da execução, acompanhamento e registro da operação diária da empresa, sendo geralmente sistemas fortemente transacionais. Exemplos: sistemas de vendas, folha de pagamento etc.;
- **Sistemas de nível de conhecimento** - suportam as pessoas que trabalham com dados e conhecimento dentro da organização. Exemplos: processadores de texto e as planilhas eletrônicas;
- **Sistemas de nível gerencial** - utilizam dados da operação e outros dados inseridos nesses sistemas para permitir a obtenção de informações que permitam a gerência, suportando a tomada de decisões, o controle e o monitoramento;
- **Sistemas de nível estratégico** - sistemas destinados a decisões de mais alto nível (efeito estratégico) e utilizam dados de todos os sistemas anteriores, normalmente de forma agregada e processada, sendo utilizados pela alta gerência.



Figura 1 - Níveis de sistemas de informação segundo Laudon

Ainda segundo Laudon, a cada nível de sistemas de informação podemos associar um ou mais tipos de sistemas (Fig. 2):

- **Sistemas de Apoio Executivo (SAE)** - encontrados no nível estratégico, são destinados a apoiar a alta gerência em tarefas estratégicas, como o planejamento de longo prazo. Usam dados fortemente agregados, internos e externos a organização e são capazes de responder perguntas específicas ou ainda fazer projeções. Podem ser capazes de fazer simulações e ter uma interface interativa;
- **Sistemas de Apoio a Decisão (SAD)** - encontrados no nível gerencial, são utilizados pelos vários níveis de gerência e utilizam como entrada dados em pequeno volume (agregações) ou ainda bases massivas de dados previamente preparadas para permitir atividades de análise de dados. Como resposta, devem fornecer relatórios específicos, análises e decisões e respostas a perguntas *Ad hoc* (*Ad hoc* significa "para esta finalidade", "para isso" ou "para este efeito". É uma expressão latina, geralmente usada para informar que determinado acontecimento tem caráter temporário e que se destina para aquele fim específico);
- **Sistemas de Informação Gerencial (SIG)**, também encontrados no nível gerencial, são utilizados pelos vários níveis de gerência. Utilizam grande volume de dados ou sumários de transações e modelos simples para obter relatórios sumários (agregados) e de exceções;
- **Sistemas de Trabalho com Conhecimento (STC)** - encontrados no nível de conhecimento, utilizam projetos, especificações e bases de conhecimento em geral para produzir modelos e gráficos. Normalmente são utilizados por profissionais com nível superior;
- **Sistemas de Automação de Escritório (SE)** - encontrados no nível de conhecimento, tem como objetivo aumentar a produtividade na manipulação de dados em um escritório. Permitem a manipulação de documentos, correio eletrônico e agendas;

- **Sistemas Processamento de Transações (SPT)** - encontrados no nível operacional, tratam eventos e transações e fornecem relatórios detalhados, listas e sumários, utilizados pelos gerentes, além de documentos específicos para a transação em que são utilizados. Os SPT suportam não só a operação diária da empresa, mas também criam os dados que são mais tarde utilizados pelos outros tipos de sistemas.

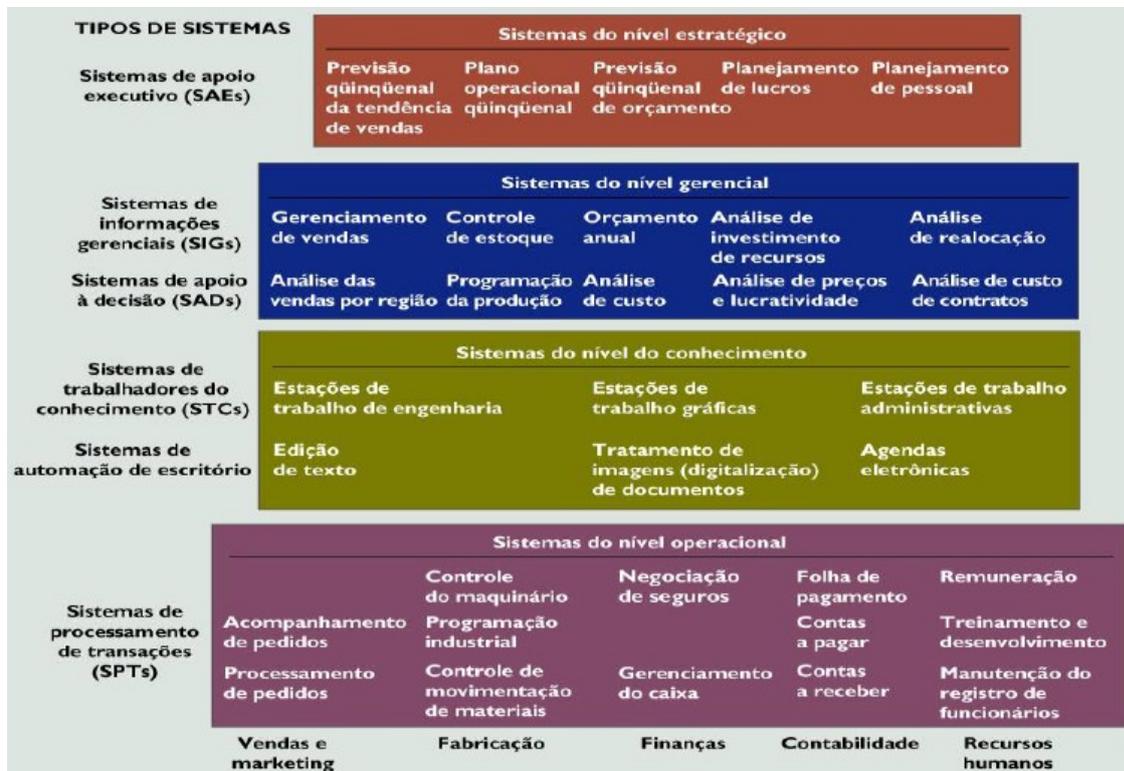


Figura 2 - Tipos de sistemas de informação segundo Laudon

1.1. Ciclo de Vida e Processo de Desenvolvimento

A norma internacional ISO/IEC 12207 fornece um framework para compreender as atividades e processos do ciclo de vida do software da sua concepção até o fim do seu uso. A norma é um referencial de mercado que estabelece os processos, as atividades e as tarefas a serem aplicadas ao longo do ciclo de vida de desenvolvimento de software.

Este referencial fornece uma estrutura comum de processos para ser utilizada como referência no desenvolvimento de produtos e serviços de software, no contexto da sistematização e gestão dos processos de trabalho de apoio ao Ciclo de Vida do Software. No entanto, não especifica os seguintes elementos: detalhes de implementação, detalhes de documentação, qual o modelo de ciclo de vida, ou o método de desenvolvimento de software.

A norma define vários processos do ciclo de vida do software e organiza-os em três classes: Processos Primários, Processos de Suporte e Processos Organizacionais (Fig. 3).

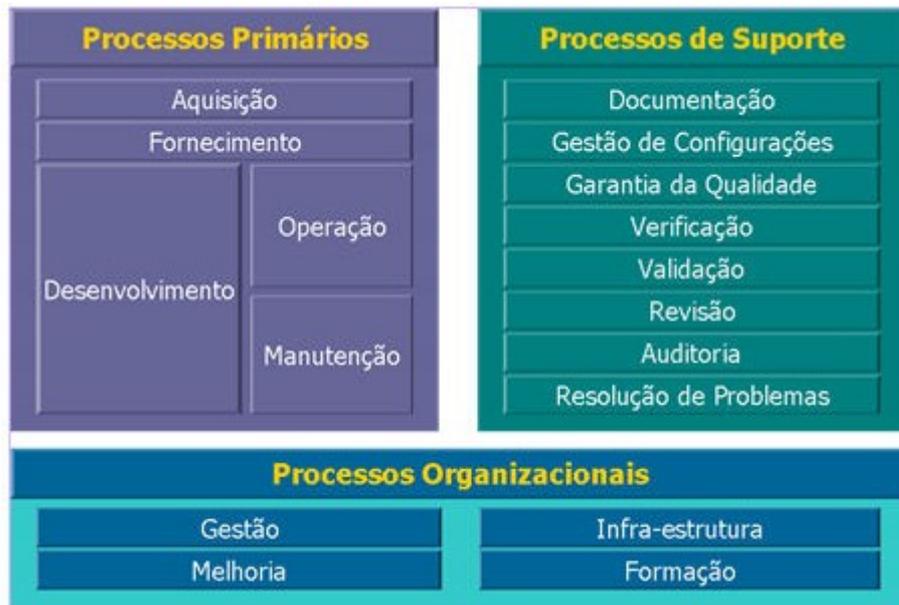


Figura 3 - Framework fornecido pela norma ISO 12207

É importante perceber a quantidade de processos envolvidos no ciclo de vida do software, pois eles nos mostram que muitas vezes subestimamos as verdadeiras necessidades para implantar e manter um sistema de informação. Nos processos primários estão enquadrados os clientes, fornecedores, operadores, usuários e a equipe de desenvolvimento. Já a equipe de suporte está inserida nos processos de suporte e organizacionais.

1.1.1. Processos Primários

Os processos primários do ciclo de vida do software são os seguintes:

- **Processo de Aquisição** - define as atividades de aquisição da organização - a organização que adquire um sistema, produto de software ou serviço;
- **Processo de Fornecimento** - define as atividades do fornecedor da organização - a organização que fornece o sistema, o produto de software ou o serviço do software ao cliente;
- **Processo de Desenvolvimento** - define as atividades da equipe de desenvolvimento da organização - a organização que define e desenvolve o produto de software;
- **Processo de Operação** - define as atividades de operação da organização - a organização que fornece o serviço de operar um sistema;
- **Processo de Manutenção** - define as atividades de manutenção do software da organização - a organização que fornece o serviço para manter o produto de software e gerir as modificações para o manter atualizado e operacional.

1.1.2. Processos de Suporte

Os processos de suporte do ciclo de vida do software são:

- **Processo de Documentação** - define as atividades para registar a informação produzida pelo processo do ciclo de vida;

- **Processo de Gestão de Configurações** - define as atividades de gestão de configuração;
- **Processo de Garantia da Qualidade** - define as atividades para assegurar de forma objetiva que os produtos e os processos de software estão em conformidade com os requisitos especificados e conforme o planejado. As revisões conjuntas, as auditorias, a verificação, e a validação podem ser usadas como técnicas da garantia de qualidade;
- **Processo de Verificação** - define as atividades (para o cliente, o fornecedor, ou parte interessada) para verificar os produtos de software;
- **Processo de Validação** - define as atividades (para o cliente, o fornecedor, ou parte interessada) para validar os produtos de software;
- **Processo de Revisão** - define as atividades para avaliar o estado e os produtos de uma atividade;
- **Processo de Auditoria** - define as atividades para determinar a conformidade com os requisitos, planejamento e contrato;
- **Processo de Resolução de problemas** - define um processo para analisar e resolver problemas (incluindo não conformidades), qualquer que seja a sua natureza ou fonte detectadas no decorrer do desenvolvimento, operação, manutenção ou outro processo.

1.1.3. Processos organizacionais

São quatro os processos organizacionais do ciclo de vida:

- **Processo de Gestão** - define as atividades básicas de gestão, incluindo a gestão de projeto, durante o ciclo de vida;
- **Processo de Infraestrutura** - define as atividades básicas para estabelecer a infraestrutura;
- **Processo de Melhoria** - define as atividades básicas ao desempenho da organização (quer seja cliente, fornecedor, equipes de desenvolvimento e manutenção ou gestor de outro processo) para estabelecer a medição, controle e melhoria do ciclo de vida;
- **Processo de Formação** - define as atividades para proporcionar a formação adequada aos colaboradores.

2. Tipos de Projetos de Sistemas de Informação

Existem três tipos de projeto de sistemas de informação: manual, manual para automático e re-automação. Os processos de re-automação ainda podem se dividir em recodificação, re-projeto e re-desenvolvimento, melhoria ou manutenção.

Todos esses tipos de projeto apresentam ao analista de sistemas o desafio de descobrir o que deve ser feito. Porém, cada tipo apresenta certas particularidades que facilitam ou dificultam esse trabalho de análise.

2.1. Sistemas manuais

O trabalho do analista em sistemas manuais é mais relacionado à formalização, por meio de documentação e padrões, de processos já adotados, a criação de novos processos e a transformação de processos existentes tendo em vista otimizá-los ou possibilitar que atendam novas necessidades da organização. Esses processos podem ser bastante complexos e de difícil entendimento em alguns casos, o que exige do analista uma boa capacidade de compreensão e modelagem. Porém,

como não serão transformados em programas de computador, o analista pode trabalhar com ferramentais mais informais e mais próximas ao dia a dia do usuário.

2.2. Manual para automático

São os projetos que apresentam maior dificuldade. Isso acontece porque esses projetos exigem todo o trabalho feito no tipo manual e, adicionalmente, a criação de um modelo computacional com certo grau de formalidade, que possa ser usado pelos desenvolvedores. Não há um guia que indique a adequação da automação ou que novos resultados podem ser obtidos. O usuário, por não ter acesso a sistemas de informação que executem a mesma atividade, tem pouco conhecimento sobre o que é possível fazer, ou não tem ideia de qual o custo de produzir certos resultados.

2.3. Re-automatização

Os projetos de re-automatização apresentam a vantagem de possuir uma base que pode ser utilizada como referência do que deve ser feito (repetido), do que não deve ser mantido e das novas atividades necessárias. O usuário, acostumado e experiente com um sistema existente, pode fornecer informações mais adequadas sobre o que espera do novo sistema, ou da manutenção ou melhoria sendo feita.

3. Desenvolvimento de Software

Desenvolver softwares é a atividade de criar um ou mais programas ou sistemas de computador, de forma a atender necessidades específicas de um cliente ou grupo de clientes. No desenvolvimento de software são realizadas atividades de descoberta das necessidades e de criação do produto de software propriamente dito.

Podemos dividir as atividades do processo de desenvolvimento de software em alguns grandes grupos:

- Atividades de Análise, para descobrir “o que” deve ser feito;
- Atividades de Projeto, para descobrir “como” o software deve ser feito;
- Atividades de Implementação, para produzir o software de acordo com as especificações produzidas nas fases anteriores;
- Atividades de Controle de Qualidade, que incluem todas as atividades com objetivo de garantir a qualidade do produto, como testes e verificações.

De acordo com o processo de desenvolvimento escolhido, cada uma dessas atividades pode ser dividida em várias outras subatividades ou tarefas. Elas podem ser executadas de diferentes formas, em diferentes ordens. Também é possível que as atividades de análise e projeto sejam feitas de forma implícita, por exemplo, quando desenvolvemos o software unicamente por meio de protótipos.

Enquanto um produto para um único usuário pode ser feito por meio de protótipos, sem nenhuma fase bem-definida, produtos maiores, como um sistema de informações gerencial, necessitam ser realizados em vários passos planejados.

4. Análise de Software

Análise é a tarefa de levantar e descrever os requisitos de um sistema, definindo de que forma deve funcionar para atender as expectativas de todos que nele possuem

algum interesse. Normalmente se diz que a análise define “o que” o sistema deve fazer sem especificar “como” fará.

Durante a análise devem ser explicitadas que tarefas o sistema deve executar e que dados deve manter em memória. Para isso, são criados um ou mais modelos do sistema, descrevendo-o com diferentes perspectivas e graus de detalhe. Ela é, simultaneamente, um acordo entre os desenvolvedores e seus clientes e um mecanismo de comunicação entre os desenvolvedores. Em ambos os casos, a análise define que serviços devem ser fornecidos pelo sistema a ser implementado e quais serviços não estão no escopo do sistema.

Segundo Pressman, todos os métodos de análise devem ser capazes de suportar cinco atividades:

1. Representar e entender o domínio da informação;
2. Definir as funções que o software deve executar;
3. Representar o comportamento do software em função dos eventos externos;
4. Separar os modelos de informação, função e comportamento de maneira a apresentar os detalhes de forma hierárquica;
5. Prover a informação essencial em direção à determinação dos detalhes de implementação.

Importante observar que, para a análise de um sistema ser útil e de qualidade, não basta entender “o que” deve ser feito, mas também desenvolver uma representação que permita documentar e comunicar essa informação.

4.1. Modelos de Análise

Vários autores fizeram propostas de modelos para descrever a análise de sistemas. Dentro do conceito de análise, apresentaremos os seguintes modelos:

- Modelo de Negócio, que descreve como funciona o negócio onde o sistema está inserido;
- Modelo de Dados, que descreve os dados guardados pela memória do sistema, na forma de um Modelo Conceitual e segundo o método de entidades e relacionamentos;
- Modelo Funcional, que descreve a funcionalidade essencial do sistema, utilizando diagramas de fluxo de dados.

4.2. Ferramentas da Análise

A maior parte do trabalho de análise é feita da comunicação entre pessoas. No desenvolvimento de sistemas se deve evitar ao máximo as ambiguidades. Para isso, é necessário restringir a linguagem utilizada de forma que uma sentença só tenha uma interpretação possível (ou mais provável). Para isso foram criadas várias linguagens, algumas gráficas, que tem o poder de restringir as interpretações possíveis do que se quer dizer.

5. Análise de Requisitos

Requisito é uma exigência, solicitação, desejo ou necessidade. A análise de Requisitos e a especificação do software servem de base para todas as atividades de Engenharia de Software.

A principal tarefa de um analista é descobrir o que o sistema deve fazer e como deve se comportar segundo as expectativas de seus usuários e outros interessados. Isso é chamado de “requisitos do sistema”. Descobrir os requisitos de um sistema é uma tarefa investigativa, criativa e contínua (Fig. 4).



Figura 4 - Análise de requisitos

O objetivo da análise é capturar todos os requisitos para o software em desenvolvimento e apenas aqueles requisitos verdadeiros. Qualquer que seja o sistema, existem várias formas de entendê-lo. Similarmente, existem vários tipos de requisitos, que são aplicáveis ou não, dependendo da visão necessária naquele instante.

Muitas vezes o analista se depara com a necessidade de, antes de definir requisitos, definir propriamente qual o problema a ser tratado. Para isso, deve fazer com que os usuários cheguem a um acordo sobre qual é o verdadeiro problema, ou o problema mais importante. Neste contexto, um requisito deve ter as seguintes características:

- **Necessário** - se retirado, haverá uma deficiência no sistema, isto é, ele não atenderá plenamente as expectativas do usuário. Em especial, não devem existir requisitos do tipo “seria interessante ter”. Ou o requisito é necessário ou é dispensável. Também deve ser levado em conta que cada requisito aumenta a complexidade e o custo do projeto, logo, não podem ser introduzidos de forma espúria;
- **Não-ambíguo** - possui uma e apenas uma interpretação. Nesse caso é muito importante atentar para a linguagem utilizada;
- **Verificável** - não sendo vago ou geral e sendo quantificado de uma maneira que permita a verificação de uma das seguintes formas: inspeção, análise, demonstração ou teste;
- **Conciso** - cada requisito define apenas o que deve ser feito de maneira clara e simples. Um requisito para ser conciso não inclui explicações, motivações, definições ou descrições do seu uso. Estes textos podem ser mantidos em outros documentos, apontados pelo requisito (de preferência sob o controle de um sistema de gerência de requisitos);
- **Independente da Implementação** - o requisito define o que deve ser feito, mas não como. Um requisito não deve refletir um projeto ou uma implementação e não deve descrever uma operação. Requisitos de interface são uma exceção a essa regra;
- **Alcançável** - ser realizável a um custo definido por uma ou mais partes do sistema;
- **Completo** - não precisa ser explicado ou aumentado, garantindo capacidade suficiente do sistema;

- **Consistente** - não contradiz ou mesmo duplica outro requisito e utiliza os termos da mesma forma que outros requisitos;
- **Ordenável** - por estabilidade ou importância (ou ambos);
- **Aceito** - pelos usuários e desenvolvedores.

Requisitos são originários das necessidades dos usuários e *stakeholders*. Stakeholders são todos aqueles com algum interesse no sistema, afetando ou sendo afetados por seus resultados. A palavra é uma composição dos termos “*stake*”, interesse ou aposta, e “*holder*”, possuidor. Um stakeholder é uma pessoa que possui algum interesse no sistema, em especial um interesse que envolve algum risco.

Enquanto os requisitos estão no universo das soluções, as necessidades vivem no “mundo dos problemas”. Os requisitos implementam as características observadas do sistema, que atendem as necessidades (Fig. 5). Um requisito é uma sentença identificando uma capacidade, uma característica física ou um fator de qualidade que limita um produto ou um processo, onde:

- **Requisito do usuário** - comportamento ou característica que o usuário deseja do software ou o sistema como um todo, ou seja, o que o usuário quer. São escritos pelo próprio usuário ou levantados por um analista de sistemas que consulta o usuário. Exemplo: O sistema deve gerar relatórios mensais que mostrem o custo dos medicamentos prescritos por clínica durante cada mês;
- **Requisito do sistema** - comportamento ou característica exigida do sistema como um todo, incluindo hardware e software. O comportamento desejado do sistema. São normalmente levantados por engenheiros ou analistas de sistemas, refinando os requisitos dos usuários e os transformando em termos de engenharia. Exemplo: No último dia de cada mês deve ser gerado um resumo dos medicamentos prescritos por clínica durante aquele mês. Um relatório por clínica deve ser gerado, listando nome dos medicamentos, total de prescrições e o custo total. Se os medicamentos estão disponíveis em diferentes unidades de dosagem (10mg, 20mg), devem ser criados relatórios separados;
- **Requisito do software** - comportamento ou característica que é exigido do software. São normalmente levantados por analistas de sistemas.

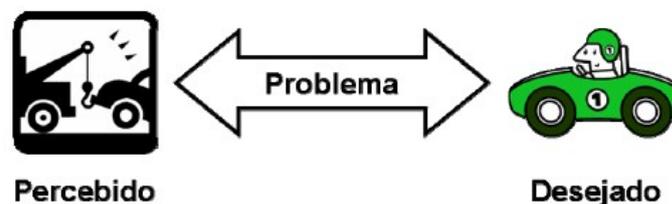


Figura 5 - Problema é uma diferença entre uma situação percebida e a desejada

5.1. Tipos de Requisitos

Os requisitos de sistemas são divididos em dois tipos: Requisitos Funcionais e Requisitos Não-funcionais.

5.1.1. Requisitos Funcionais

Quando falamos de um Requisito Funcional estamos nos referindo à requisição de uma função que um software deverá atender/realizar. Ou seja, exigência, solicitação, desejo, necessidade, que um software deverá materializar.

Um requisito funcional representa algo que o sistema deve fazer, ou seja, uma função esperada do sistema que agregue algum valor a seus usuários. Estão diretamente ligados a funções que o sistema deve fornecer, como o sistema deve reagir a entradas específicas ou como o sistema deve se comportar em determinadas situações e também podem declarar o que o sistema não deve fazer (Fig. 6).



Figura 6 - Requisitos funcionais

Requisito funcional não é uma funcionalidade, é uma necessidade funcional (uma função) que o software deve atender. Uma funcionalidade que será executada por um ator sistêmico (o próprio sistema) ou por um ator humano (usuário final). São exemplos de requisitos funcionais:

- “O usuário **deve** conseguir fazer buscas em todo o acervo de materiais bibliográficos”;
- “O sistema **deve** fornecer telas apropriadas para o usuário ler documentos disponíveis no repositório de documentos”;
- “O sistema **deve** permitir o cadastro dos fornecedores da loja”;
- “O sistema **deve** utilizar os dados obtidos a partir dos sensores e interpretá-los para realizar a navegação”.

Observar que um requisito funcional se refere a algo que será feito, uma ação a ser realizada pelo sistema. Por isso o nome precisa estar sempre no tempo verbal infinitivo. Um requisito funcional que trata do “expurgo de registros de clientes inativos” não pode ter este nome, deve ser referido como “Expurgar registros de clientes inativos”.

É comum os analistas associarem a ideia de um requisito funcional a uma tela, uma rotina, que no fim serão as funcionalidades de fato de um sistema. Entretanto, uma funcionalidade não necessariamente realizará apenas um requisito funcional. Uma funcionalidade pode realizar vários requisitos funcionais, ou seja, em uma funcionalidade, um ou mais requisitos funcionais podem ser atendidos (Fig. 7).

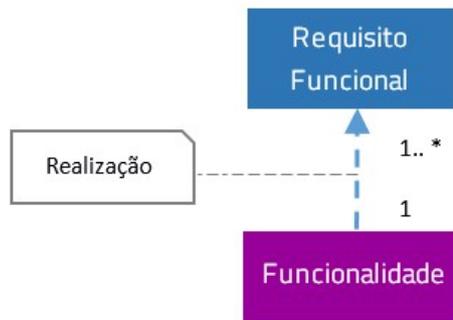


Figura 7 - Requisito funcional e funcionalidades

Por exemplo, um sistema que possui uma tela para “Manutenção de Clientes” e que mantém os dados cadastrais dos clientes no sistema. Isto representa uma única funcionalidade. Nesta tela é possível incluir/alterar/consultar/excluir clientes dos tipos pessoa física e pessoa jurídica. Mas quantos requisitos são realizados (atendidos) por esta funcionalidade? A Tabela 1 mostra os requisitos funcionais necessários para a funcionalidade exemplo:

Tabela 1 - Requisitos funcionais

Requisitos Funcionais – Identificador e Nome	
RF001	Incluir cliente pessoa física
RF002	Alterar cliente pessoa física
RF003	Consultar cliente pessoa física
RF004	Excluir cliente pessoa física
RF005	Incluir cliente pessoa jurídica
RF006	Alterar cliente pessoa jurídica
RF007	Consultar cliente pessoa jurídica
RF008	Excluir cliente pessoa jurídica

5.1.2. Precisão dos Requisitos Funcionais

Surgem vários problemas quando os requisitos não são declarados de forma precisa. Requisitos ambíguos podem ser interpretados de diferentes maneiras pelos desenvolvedores e usuários. Considere o termo ‘telas apropriadas’ do exemplo anterior:

- Intenção do cliente: telas especiais para cada tipo diferente de documento;
- Interpretação do Desenvolvedor: fornecer uma tela texto que mostra o conteúdo do documento.

Os requisitos funcionais devem ainda ser completos e consistentes:

- **Completos** - Eles devem incluir descrição de todas as facilidades que estão sendo requeridas;
- **Consistentes** - Eles não devem apresentar conflitos ou contradições entre as descrições das facilidades fornecidas pelo sistema.

Observar que na vida real é impossível produzir um documento de requisitos que seja absolutamente completo e consistente.

5.1.3. Qualidade dos Requisitos Funcionais

Um Requisito Funcional de qualidade precisa atender alguns atributos específicos. Na literatura existem recomendações de atributos que um requisito deve atender para ter qualidade. A Tabela 2, apresenta alguns atributos e suas características.

Tabela 2 - Atributos e características para requisitos funcionais

Atributo	Características
Unidade	Deve ter uma única proposta. Não deve atender a mais de uma exigência. “Incluir cliente” não é unitário, pois se refere a incluir clientes de tipos diferentes (pessoa física e jurídica), assumindo assim várias responsabilidades, quando deveria assumir apenas uma.
Compleitude	Deve ser autocontido, deve ter “início/meio/fim”, ser completo. “Pagar fatura” não é completo, só conta é “parte da história”. Para ser completo deveria ser algo como “Pagar fatura com cartão de crédito para cliente pessoa física”.
Consistência	Não deve contradizer outro requisito funcional do mesmo escopo do projeto. É como termos dois requisitos propondo fazer a mesma coisa, mas cada um propondo fazer de formas diferentes.
Atomicidade	Para ser atômico o requisito funcional precisa ter unidade. Atomicidade remete a assumir apenas uma responsabilidade. Também deve ser algo indivisível, não podendo ser decomposto. Requisitos funcionais dependem de outros para se efetivarem. “Realizar compra de produto” e “Realizar pagamento com cartão de crédito” apresenta um único requisito funcional que é “Realizar compra de produto com pagamento em cartão de crédito”.
Não-Ambiguidade	Não pode ser ambíguo, não pode propor algo que não seja claro. “Emitir relatório” não quer dizer nada. Relatório de que, para que? “Emitir relatório de saldo” é um pouco mais completo. Saldo de que? “Emitir relatório de saldo da conta corrente do cliente pessoa física” é um requisito funcional não-ambíguo.
Verificável	Deve ser possível de associar com um artefato de construção, de testes. Um requisito funcional deve ser testável e passível de entendimento como foi construído e homologado. Para isso tem que ser rastreável.
Rastreável	Deve ser localizável no sistema terminado, funcional e executável. Como saber se um requisito funcional foi atendido? Para isso é necessário ter rastreabilidade e isso só é possível associando à interface gráfica, que será associada a um caso de uso, que será associado a funcionalidades, que serão implementadas etc.
Prioridade	Um requisito funcional essencial é algo muito diferente de um requisito funcional desejável; ambos possuem valores completamente diferentes para o negócio. O requisito funcional deve possuir prioridade, isso interfere diretamente no projeto do software.

5.2. Requisitos Não-Funcionais

Requisitos não funcionais tratam da forma como os requisitos funcionais devem ser alcançados. Eles definem propriedades e restrições do sistema. Muitos requisitos não funcionais são também requisitos de qualidade, como exigências de desempenho e robustez. Outros são restrições ou exigências de uso de uma ou outra tecnologia. Em geral, são requisitos que expressam restrições que o software deve atender e as qualidades específicas que o software deve ter. A Figura 8, apresenta os tipos de requisitos não funcionais:

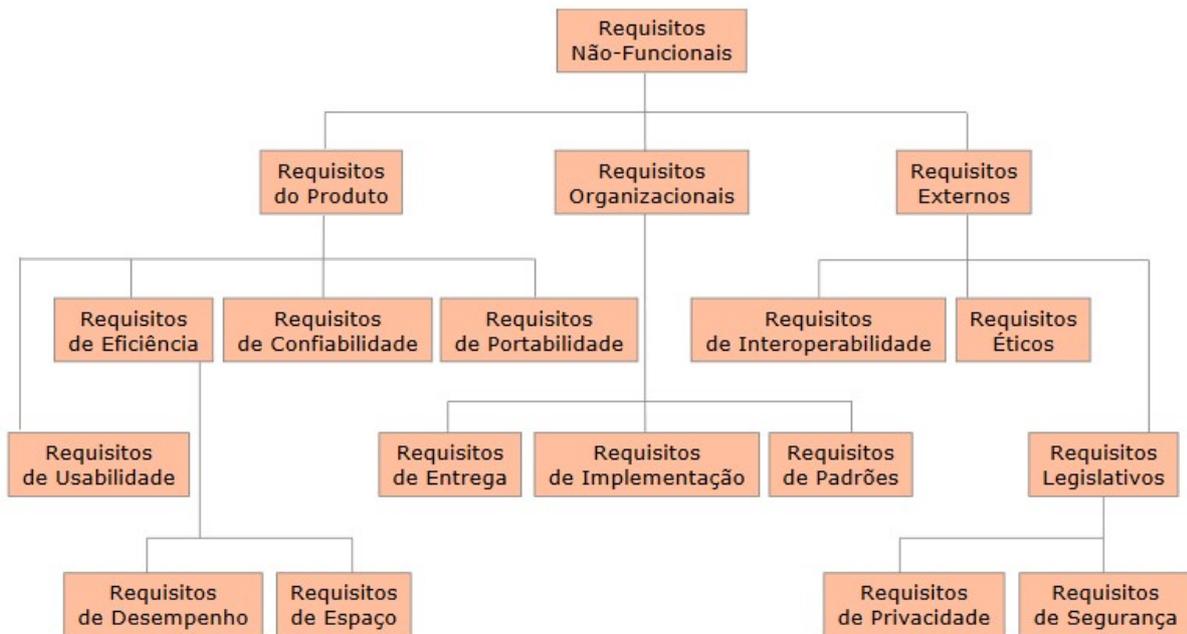


Figura 8 - Requisitos não-funcionais

Exemplos

- Requisitos do Produto - o sistema deve ser robusto e tolerante a falhas, de forma a continuar sua operação ou abortar de forma segura o modo autônomo caso haja falha de um ou mais sistemas essenciais;
- Requisitos Organizacionais - o processo de desenvolvimento do sistema e os produtos liberáveis devem estar em conformidade com o padrão empresarial XYZ;
- Requisitos Externos - os usuários do sistema não devem ter acesso a qualquer dado que não necessitem.

5.2.1. Metas dos Requisitos Não-Funcionais

Os requisitos não-funcionais podem ser muito difíceis de serem declarados precisamente. Para facilitar isso podem ser utilizadas metas que transmitam as intenções dos usuários do sistema. Por exemplo, um sistema de controle de aeronave deve ser fácil de ser usado por controladores experientes e deve estar organizado de tal maneira que os erros dos usuários sejam minimizados.

5.2.2. Interação entre Requisitos Não-Funcionais

Em sistemas complexos são comuns conflitos entre diferentes requisitos não-funcionais. Seguindo o exemplo anterior, no sistema para aeronaves, para minimizar o peso, o número de chips do sistema deve ser minimizado e para minimizar o consumo de energia, devem ser usados os chips de menor potência. Entretanto, usar chips de menor potência pode significar que mais chips devem ser usados. Qual é o requisito mais crítico?

5.3. Outros tipos de Requisitos

5.3.1. Casos de Uso

Uma outra forma de especificar requisitos é através de Casos de Uso, que contém o nome da funcionalidade e pode ser conter sua respectiva descrição. Um diagrama de Casos de Uso auxilia no levantamento dos requisitos funcionais do sistema, descrevendo um conjunto de funcionalidades do sistema e suas interações com elementos externos e entre si. Os diagramas de Casos de Uso apresentam como características:

- Auxiliam na comunicação entre o cliente e os analistas;
- Apresentam as principais funcionalidades do sistema com foco no cliente;
- Descrevem cenários de interação entre as partes internas/externas de um sistema, com foco no usuário;
- Muito utilizados na fase de levantamento de requisitos.

No Caso de Uso, o analista define o ator de maneira a relacionar com as funcionalidades, a partir dessa fase fica mais bem descrito os requisitos para então iniciar a modelagem do banco, desenvolvimento e outras etapas.

Por exemplo, uma loja on-line, onde temos o cenário "comprar produto". O cliente navega pelo catálogo e adiciona os itens desejados ao carrinho de compras. Quando o cliente desejar pagar, ele preenche os dados para entrega e da forma de pagamento, confirmando a compra no final. O sistema verifica o cartão de crédito, tenta passar a compra no cartão e confirma a venda com o envio de um e-mail (Fig. 9).

Este é um de vários cenários que podem ocorrer, tendo em vista que a verificação do cartão pode dar não autorizado, gerando um segundo cenário. Seu cliente pode ser um cliente regular, não necessitando preencher os dados de entrega ou mesmo do cartão, isso já seria um terceiro cenário.

A essência desses cenários é que o usuário tem o mesmo objetivo: comprar um produto; nem sempre o usuário consegue, mas o objetivo permanece. Então, um caso de uso é um conjunto de cenários relacionados por um objetivo comum a um usuário.

Os cenários devem se concentrar na intenção do ator, não na mecânica de como o ator utilizará o sistema. Por isso, detalhes da implementação da interface de usuário não devem ser descritas nesse momento.

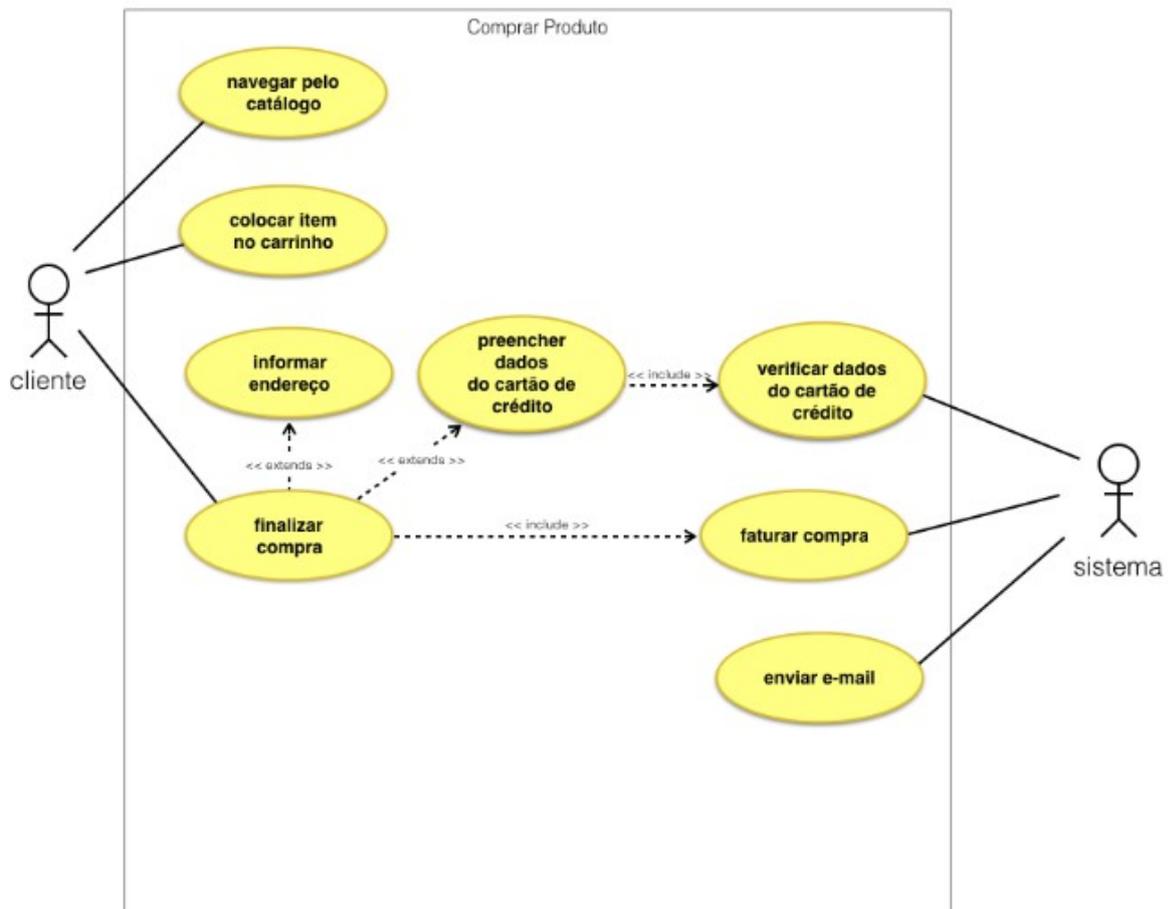


Figura 9 - Diagrama de Caso de Uso

No exemplo acima existem dois atores no diagrama, o cliente e o sistema. Os casos de uso do sistema são abstratos, pois dependem de o cliente interagir com seus casos de uso para que os casos de uso do sistema possam ser acionados.

James e Susan Robertson, no *Atlantic System Guild*, identificam mais três tipos de requisitos: restrições de projeto, temas de projeto e impulsionadores de projeto:

- **Restrições de projeto** - são os limites impostos ao sistema para que funcione no seu ambiente operacional. Estas restrições tanto podem ser técnicas, envolvendo necessidades ou disponibilidades de fatores como hardware, software, rede e interoperabilidade com outros sistemas, quanto podem ser ligadas ao negócio. Restrições podem ser vistas como requisitos não funcionais de certa forma especiais, pois são limitantes;
- **Impulsionadores de projeto** - são as forças do negócio que fazem o projeto acontecer. Podem ser considerados requisitos na medida em que são os geradores originais dos requisitos funcionais e não funcionais. Tanto o objetivo inicial do sistema como todos os nele interessados (stakeholders) são impulsionadores;
- **Assuntos de projeto** - servem para completar o quadro geral de todos os fatores que influenciarão o sucesso ou o fracasso do projeto.

5.4. Requisitos Verdadeiros e Falsos

Um requisito é verdadeiro quando o sistema deve cumpri-lo qualquer que seja a tecnologia de implementação escolhida. Se um sistema pode cumprir sua finalidade sem que um requisito seja implementado, então esse requisito é falso. Qualquer requisito que não possa ser verificado, ou seja, que sua presença não possa ser garantida por alguma forma quantificável e mensurável no final do projeto é um requisito falso.

Requisitos falsos aparecem de várias formas dentro do sistema: por cópia de sistemas antigos, por hábito do analista, por pensarmos na tecnologia antes do tempo. Dividem-se em dois grupos:

- **Requisitos falsos tecnológicos** - são aqueles incluídos em um sistema por antecipação de alguma característica tecnológica futura, como a linguagem de programação a ser utilizada, ou por alguma característica tecnológica passada, como o tipo de arquivo em que estão guardados os dados atualmente. Podem ser divididos em:
 - Requisitos tecnológicos incluídos pelo passado, quando incluímos na especificação algo que existe na implementação existente, mas que não é necessário ao funcionamento do sistema;
 - Requisitos tecnológicos incluídos por antecipação, quando incluímos algo na especificação em função de alguma tecnologia escolhida para a implementação;
- **Requisitos falsos arbitrários** - são incluídos pelos analistas por preciosismo ou por características da ferramenta de modelagem utilizada. Podem ser divididos em:
 - Requisitos arbitrários, incluídos por influência da ferramenta de modelagem, quando incluímos na especificação algo desnecessário para fazer o sistema, mas necessário por alguma característica da ferramenta de modelagem;
 - Requisitos arbitrários incluídos por preciosismo, quando incluímos uma função espúria no sistema, por exemplo, uma função que foi não solicitada pelo usuário.

5.5. Documentação dos requisitos

Requisitos de projeto podem ser descritos com as seguintes informações:

- Número identificador - para facilitar a discussão, identificam-se todos os requisitos unicamente.
- Tipo - classificando-o como funcional, não funcional etc.;
- Evento que o atende;
- Descrição;
- Justificativa;
- Fonte do requisito - a pessoa ou o grupo que o originou;
- Critério de aceitação - medida que possa ser usada para garantir que o requisito foi alcançado;
- Satisfação do usuário - grau, de 1 (nenhum interesse) a 5 (extremamente satisfeito), por exemplo, indicando a satisfação do cliente se esse requisito for alcançado;

- Insatisfação do usuário - grau, de 1 (nenhum interesse) a 5 (extremamente insatisfeito), por exemplo, indicando a satisfação do cliente se esse requisito não for alcançado;
- Dependências - referências a outros requisitos que dependem de alguma forma desse requisito;
- Conflitos - referência aos requisitos que de alguma forma conflitam com esse;
- Material de apoio - listagem de material de apoio para atender esse requisito;
- Histórico;
- Documentação da criação e das mudanças efetuadas.

5.6. Estrutura de um Requisito Funcional

Não há um padrão estabelecido quanto a estrutura de um requisito funcional. Boa parte das organizações utiliza um formato contendo campos específicos. O modelo da Tabela 3, apresenta os campos mais relevantes, com posterior descrição de cada um.

Tabela 3 - Modelo de requisição funcional

Identificador			
Nome			
Módulo			
Data de criação		Autor	
Data da última alteração		Autor	
Versão		Prioridade	
Descrição			

- **Identificador** - Sufixo seguido de um identificador único. O sufixo geralmente utilizado é RF (Requisito Funcional) e o identificador único geralmente é composto de quatro dígitos. O uso de identificador é fundamental para uma melhor organização do projeto. Localizar requisitos pelo nome é pouco prático devido ao volume dos requisitos e ambiguidades nos nomes. Também é importante algum recurso de numeração automática, pois controlar os números dos identificadores manualmente é inviável;
- **Nome** - Nome curto do RF, mas que possibilite entender o que ele faz apenas por essa nomenclatura;
- **Módulo** - Módulo ao qual o RF pertence. Se for um sistema pequeno que não possua nenhum módulo (somente o próprio sistema), deve ser preenchido com N/A (não se aplica).
- **Data de criação / Autor** - Data da criação do RF, ou a data em que ele foi especificado. Autor é o profissional que especificou o RF pela primeira vez (quem o criou);
- **Data da última alteração / Autor** - Data em que houve a última alteração no RF. Autor – é o profissional que alterou a especificação do RF pela última vez;
- **Versão** - Número da versão do RF. Geralmente utiliza-se algo simples, como 1, 2. A versão inicial sempre é a 1, e a cada alteração incrementa-se a versão;
- **Prioridade** - Se o RF é Essencial, Importante ou Desejável.

- **Descrição** - Descrição detalhada (a mais detalhada possível) do RF.

É recomendado que se utilize, sempre que possível, alguma ferramenta Case que dê suporte a Engenharia de Requisitos, para melhorar a produtividade na modelagem e a gestão dos requisitos. Na Tabela 4, temos um exemplo de um RF especificado para a área financeira de uma empresa.

Tabela 4 - Exemplo de um Requisito Funcional especificado

Identificador	RF0001		
Nome	Emitir carta de cobrança para clientes inadimplentes conforme critérios pré-estabelecidos		
Módulo	Financeiro		
Data de criação	30/03/2020	Autor	José Pinheiro
Data da última alteração	N/A	Autor	N/A
Versão	1	Prioridade	Essencial
Descrição	<p>Para todo cliente que esteja inadimplente deve ser possível a emissão de carta de cobrança através do sistema. O resultado da emissão é a carta impressa, que será posteriormente despachada por correios ou outro agente de entrega.</p> <p>Os critérios para definir se um cliente é ou não inadimplente devem estar cadastrados no sistema utilizando regras de negócio específicas. Nem todo cliente com pagamento em atraso será considerado inadimplente, pois deverá ser levado em consideração o score de crédito do cliente, sua renda mensal, patrimônio etc. Verificar os Requisitos Funcionais e regras de negócio específicas para manutenção dos critérios citados.</p> <p>Não haverá diferença na emissão da carta de cobrança para clientes pessoa física (particular) ou jurídica (empresarial). O processo será o mesmo para ambos.</p>		

5.7. Dependência de Requisitos

Muitos requisitos só podem ser implementados se outros requisitos forem implementados antes. Por exemplo, é impossível fazer um relatório de vendas sem que se cadastre previamente as vendas no sistema.

Uma das atividades da gerência de requisitos é manter esse relacionamento de dependência, que influenciará em todo desenvolvimento e processos do sistema. Uma solução é manter uma tabela de dependência de requisitos, outra manter um banco de dados de requisitos que inclua relações de dependência. Existem produtos no mercado especializados na gerência de requisitos.

5.8. Priorização dos Requisitos

Existe a tendência de o sistema crescer muito durante a análise, principalmente no momento de entrevistar os usuários, quando acontece uma facilidade natural das pessoas para propor novas funcionalidades para um sistema que ainda não existe, por imaginarem alguma utilidade nessas funções propostas. Nesse caso, algumas técnicas podem ser utilizadas para caracterizar o que deve ser realmente feito ou, pelo menos, em que ordem as coisas devem ser feitas.

A primeira técnica é associar a cada requisito do sistema uma importância. Uma escala de três ou cinco valores é adequada para isso, como em: “Imprescindível para o sucesso do sistema”, “Funcionalidade Importante, mas podemos esperar algum tempo”, “Ajudaria ter, mas é possível viver sem essa funcionalidade”, “Benefícios mínimos”, “Desnecessário”.

A segunda técnica é planejar o sistema para ser entregue em várias versões, mesmo que nem todas as versões estejam incluídas nesse contrato, e pedir para o cliente determinar que funcionalidades devem aparecer em cada versão. Nesse caso pode ser interessante dar um peso ou custo para cada requisito, de modo que o cliente possa “controlar seus gastos”.

Uma terceira técnica é dar uma moeda virtual para o cliente, por exemplo, 1000 coins, e pedir para ele distribuir quanto pagaria por cada função, priorizando no desenvolvimento aquelas funções que o cliente decidir pagar mais.

Todas essas técnicas, porém, ficam dependentes de uma outra priorização importante dos requisitos: a priorização por dependência. Devem ser levados em conta os vários fatores que influenciam nessa determinação de prioridades, entre eles:

- Diminuir o custo da implementação;
- Valor para o comprador;
- Tempo para implementar;
- Facilidade técnica de implementar;
- Facilidade do negócio para implementar;
- Valor para o negócio;
- Obrigação por alguma autoridade externa.

6. Identificação dos Requisitos

Para o bom desenvolvimento de um software é importante identificar os requisitos, pois a partir desta fase podem surgir muitos erros, que se não corrigidos a tempo impactaram em tempo do custo de desenvolvimento e valor (Fig. 10).



Figura 10 – Identificação dos requisitos

6.1. Elicitação e Análise de Requisitos

A elicitação de requisitos é o levantamento, registro e validação das expectativas dos diversos interessados no sistema, seguido da consolidação e validação dessas expectativas em requisitos formais. Inicialmente é preciso identificar o(s) problema(s) a resolver, os envolvidos no sistema e as pessoas que direta ou indiretamente influenciam no produto (sistema/app), ou seja, quem vai utilizar e a partir daí coletar as informações preliminares para os requisitos.

Vale ressaltar que os requisitos são fundamentais tanto para estimativa de preço quanto para modelagem, projeto, implementação, testes e manutenibilidade. Dessa forma, os requisitos tornam-se intrínsecos ao ciclo de vida do software, ou seja, como se fosse a água para a vida dos seres vivos.

Para levantar requisitos é necessária a interação entre aqueles que conhecem os requisitos ou a necessidades dos usuários e stakeholders e os desenvolvedores. É

um processo iterativo de comunicação, onde, por aproximações sucessivas, o desenvolvedor constrói um modelo aceito pelos usuários (Fig. 11).



Figura 11 - Elicitação de requisitos

6.2. Especificação de Requisitos

A especificação de um sistema ou software pode conter requisitos funcionais e não-funcionais e até mesmo um diagrama de caso de uso ou prototipação de parte do produto. São descritos o passo a passo de cada funcionalidade bem como suas devidas restrições. Os requisitos e as formas de obtê-los e documentá-los variam de um projeto para o outro. Contudo, existe uma série de atividades genéricas comuns a todos os processos.

6.3. Validação de Requisitos

Após a especificação das necessidades do usuário é importante validar esses dados, seja através de uma reunião fazendo com que os responsáveis assinem o documento para que ele possa ter validade, ou dando ciência da responsabilidade dos requisitos levantados por e-mail. Essa etapa também serve para correções e podem ser descobertos/inclusos outras funcionalidades.

6.4. Documento de Requisitos

O documento de requisitos serve para apresentar uma visão geral e funcional do produto e deve conter:

- Introdução;
- Visão geral do produto;
- Termos técnicos específicos para um determinado contexto;
- Abreviações e acrônimos;
- Envolvidos e Usuários;
- Requisitos (Funcionais, Não-Funcionais e Regras de Negócio);
- Caso de Uso;
- Anexos (protótipos, arquitetura e documentos auxiliares).

6.5. Extração de Requisitos

A extração de requisitos é o processo de transformação das ideias que estão na mente dos usuários (a entrada) em um documento formal (saída). É a fase de elicitação, ou seja, a criação de sistema solicitado pelo cliente (Fig. 12).

A meta é o reconhecimento dos elementos básicos do problema, conforme percebidos pelo cliente. Envolve pessoal técnico trabalhando com os usuários para descobrir sobre o domínio da aplicação, os serviços que o sistema deve fornecer e sobre as restrições operacionais. Pode envolver usuários finais, gerentes, pessoal técnico, especialistas de domínio, representantes sindicais etc.

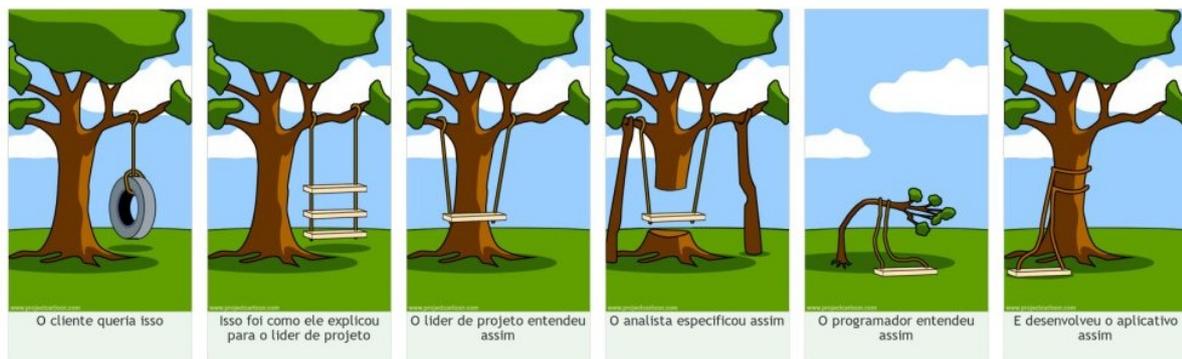


Figura 12 - Extração de requisitos

É um processo crítico no projeto de software, pois requisitos incompletos, incorretos ou mal entendidos são as causas mais frequentes da baixa qualidade, excesso de custo e atrasos nas liberações do software.

Algumas técnicas são propostas visando auxiliar a comunicação e a extração dos requisitos:

- **Entrevistas** - entrevista formal ou informal, a equipe formula questões para os stakeholders sobre os sistemas que eles usam e o sistema a ser desenvolvido. Dois tipos de entrevistas:
 - Entrevistas fechadas: um conjunto de questões predefinidas são respondidas;
 - Entrevistas abertas: não há um roteiro predefinido, uma variedade de assuntos é explorada com os stakeholders;
- **Cenários** - são exemplos reais de como o sistema pode ser usado. Por exemplo, saque em caixa eletrônico, empréstimo de livro em biblioteca, compra de livro na internet etc. Eles devem incluir:
 - Descrição da situação inicial;
 - Descrição do fluxo normal de eventos;
 - Descrição do que pode dar errado;
 - Informação sobre outras atividades concorrentes;
 - Descrição do estado quando o cenário termina;
- **Depoimentos do usuário** - São frases escritas pelo usuário na sua linguagem, sobre algo que a aplicação deve fazer. Eles conduzem a novas reuniões que podem ocorrer durante a fase de desenvolvimento. Exemplos para uma loja virtual: Um usuário possui um carrinho de compras no qual ele adiciona produtos que quer comprar. Um usuário faz o pagamento com cartão de crédito ou boleto bancário. Um usuário lê comentários feitos por

outros sobre os produtos da loja. Um usuário recebe um e-mail de confirmação de compra quando efetua um pagamento;

- **Etnografia** - Requisitos do sistema se originam do modo como as pessoas realmente trabalham. O analista deve observar e analisar como as pessoas trabalham. Estudos de etnografia mostram que o trabalho é, geralmente, mais rico e mais complexo do que o sugerido pelos modelos simples de sistema e fatores sociais e organizacionais de importância podem ser observados. Ideal complementar com prototipação;
- **Brainstorm** – ou tempestade de ideias, é uma técnica realizada em grupo para gerar ideias relativas a um determinado assunto. É útil na geração de uma ampla variedade de pontos de vista sobre o problema e na formulação do mesmo de diferentes maneiras. Útil no início do processo de extração de requisitos e permite que pessoas sugiram e explorem ideias sem que sejam criticadas e julgadas. Funciona melhor com um número mínimo de quatro e máximo de dez pessoas. Pode ser dividida em duas fases:
 - Fase de Geração: participantes são encorajados a fornecer ideias, sem discussão quanto ao mérito das mesmas;
 - Fase de Consolidação: as ideias são discutidas, revisadas e organizadas;
- **Prototipação** - Construção de modelos que representam o software. Protótipos executáveis (papel ou software).

Os erros mais comuns cometidos no desenvolvimento envolvem:

- Ignorar um grupo de clientes;
- Ignorar um único cliente;
- Omitir um grupo de requisitos;
- Permitir inconsistências entre grupos de requisitos;
- Aceitar requisito inadequado;
- Aceitar requisito incorreto, indefinido ou impreciso;
- Aceitar um requisito ambíguo e inconsistente.

Exercícios

1. Identifique os requisitos funcionais e não funcionais. Aponte possíveis incertezas na descrição.

Um sistema automático de emissão de passagens vende passagens de trem. A partir de uma lista de possíveis destinos, os usuários escolhem seu destino e apresentam um cartão de crédito e um número de identificação pessoal. Os destinos possíveis devem ser organizados de modo a facilitar a escolha. Após a escolha do destino, o sistema deve responder prontamente se há espaço disponível no trem. A passagem é emitida e o custo dessa passagem é incluído em sua conta do cartão de crédito. Quando o usuário pressiona o botão para iniciar, uma tela de menu com os possíveis destinos é ativada, juntamente com uma mensagem para que o usuário selecione um destino. Uma vez selecionado um destino, pede-se que os usuários insiram seu cartão de crédito. A validade do cartão é checada e o usuário então deve fornecer um número de identificação pessoal. Quando a transação de crédito é validada, a passagem é emitida. O formato do bilhete de passagem deve seguir ao padrão definido pelo Sistema Nacional de Tráfego Ferroviário.

2. Aponte possíveis incertezas na descrição abaixo.

Um sistema automático de emissão de passagens vende passagens de ônibus. Os usuários escolhem seu destino e apresentam um cartão de crédito e um número de identificação pessoal. A passagem é emitida e o custo dessa passagem é incluído em sua conta do cartão de crédito. Quando o usuário pressiona o botão para iniciar, uma tela de menu com os possíveis destinos é ativada, juntamente com uma mensagem para que o usuário selecione um destino. Uma vez selecionado um destino, pede-se que os usuários insiram seu cartão de crédito. A validade do cartão é checada e o usuário então deve fornecer um número de identificação pessoal. Quando a transação de crédito é validada, a passagem é emitida. O formato do bilhete de passagem segue o padrão definido pelo órgão responsável de controle rodoviário.