

Engenharia de Requisitos e Processos de *Software*



Cruzeiro do Sul Virtual
Educação a distância

Material Teórico



Fundamentos de Engenharia de Requisitos

Responsável pelo Conteúdo:

Prof. Me. Artur Marques

Revisão Textual:

Prof. Me. Luciano Vieira Francisco

UNIDADE

Fundamentos de Engenharia de Requisitos



- Engenharia de Requisitos;
- Engenharia de Requisitos de *Software*;
- Requisitos do Sistema;
- Análise de Problemas – Requisitos;
- Classificação de Requisitos;
- Partes e Questionamentos em uma Especificação de Requisitos de *Software*.



OBJETIVO DE APRENDIZADO

- Conhecer os conceitos da engenharia de requisitos para a construção de *software*.



Orientações de estudo

Para que o conteúdo desta Disciplina seja bem aproveitado e haja maior aplicabilidade na sua formação acadêmica e atuação profissional, siga algumas recomendações básicas:

Determine um horário fixo para estudar.

Mantenha o foco! Evite se distrair com as redes sociais.

Procure manter contato com seus colegas e tutores para trocar ideias! Isso amplia a aprendizagem.

Seja original! Nunca plágie trabalhos.

Aproveite as indicações de Material Complementar.

Conserve seu material e local de estudos sempre organizados.

Não se esqueça de se alimentar e de se manter hidratado.

Assim:

- ✓ Organize seus estudos de maneira que passem a fazer parte da sua rotina. Por exemplo, você poderá determinar um dia e horário fixos como seu “momento do estudo”;
- ✓ Procure se alimentar e se hidratar quando for estudar; lembre-se de que uma alimentação saudável pode proporcionar melhor aproveitamento do estudo;
- ✓ No material de cada Unidade, há leituras indicadas e, entre elas, artigos científicos, livros, vídeos e sites para aprofundar os conhecimentos adquiridos ao longo da Unidade. Além disso, você também encontrará sugestões de conteúdo extra no item **Material Complementar**, que ampliarão sua interpretação e auxiliarão no pleno entendimento dos temas abordados;
- ✓ Após o contato com o conteúdo proposto, participe dos debates mediados em fóruns de discussão, pois irão auxiliar a verificar o quanto você absorveu de conhecimento, além de propiciar o contato com seus colegas e tutores, o que se apresenta como rico espaço de troca de ideias e de aprendizagem.

Engenharia de Requisitos

A engenharia de requisitos preocupa-se em entender um sistema – *software* ou não – que o cliente tenha solicitado. Fornece a base na qual o *design* e desenvolvimento de um produto ou *software* pode prosseguir. É importante ressaltar que se os criadores/desenvolvedores não entenderem adequadamente os requisitos, é provável que o resultado não atenda às necessidades do cliente. Isso torna a compreensão dos requisitos do cliente importantes ao sucesso do projeto de desenvolvimento.

Como exemplo, você pode pensar em um veículo que apresente uma série de problemas de *design*, mecânica, consumo e conforto, e que não atende às necessidades de seus donos, tendo fracassado exatamente porque falhou na compreensão do que estes clientes, em especial, queriam. Por outro lado, quando pretendemos fabricar um *software*, é de uma importância decisiva entender o que o cliente necessita para produzir um artefato de *software* capaz de proporcionar o tipo de serviço adequado ao que foi solicitado, levando em consideração outros fatores, já que o **tempo é curto**, normalmente por praxe de mercado, e **os valores, altos**. Portanto, *software* bom é *software* com requisitos bem feitos e, principalmente, bem compreendidos.

Portanto, a engenharia de requisitos **concentra-se em descobrir o que deve ser desenvolvido – e não como deve ser desenvolvido**.

Engenharia de Requisitos de *Software*

A engenharia de requisitos é o **processo de adequação dos projetos** a um conjunto de requisitos básicos de *software*. Isto é extremamente importante para criar resultados precisos em engenharia de *software*.

Consiste em um conjunto de atividades cujo propósito é identificar e comunicar a finalidade de um sistema de *software* e os contextos em que serão utilizados. Portanto, atua como ponte entre as necessidades reais de usuários, clientes e outras partes interessadas e afetadas por um sistema de *software*, e as capacidades e oportunidades oferecidas por tecnologias intensivas exploradas através dos *softwares*.

Na engenharia de requisitos de *software* é analisado um conjunto de dados referentes às metas e aos objetivos do *software*: como funcionará? Quais são as qualidades e propriedades que deve ter para fornecer os resultados necessários? Trabalhamos a partir destas questões e destes dados para analisar diferentes soluções de construção que suportam e propiciam esses resultados.

Quando mencionamos metas e objetivos, temos por trás de tudo isso o que chamamos de **propósito do software!**

Há, dentro da indústria de requisitos, isto que chamamos atualmente de **complexidade de propósito**. O *design* de sistemas intensivos de *software* pertence a uma classe de problemas conhecidos como **heréticos** – em referência à dificuldade de se definir bem a qual domínio pertence o problema em questão: atualmente, há muitas áreas “cinentas” em *software*, e outras tantas estão contidas nas pessoas e formas de clientes ou de usuários. Quando dizemos “herético”, referimo-nos às seguintes situações:

- Não há formulação definitiva do problema, porque diferentes partes interessadas têm, cada uma, a sua própria concepção do problema;
- Não há nenhuma regra de parada, porque cada solução é suscetível de levar a novas percepções sobre o problema, e o problema nunca é suscetível de ser resolvido inteiramente. Por isso, após o término de um projeto de *software*, há tantas reuniões, tantos pedidos de partes interessadas, solicitando saber o que será feito no futuro para complementar a versão entregue do *software*. Por isso também tanto **feature road map** ou, se preferir, **mapas de entrega de funcionalidades**, de tempos em tempos, permitindo ao usuário saber, com antecedência, o que será entregue, para saber se agora será atendido;
- As soluções não são certas ou erradas, mas simplesmente melhores ou piores;
- Não há nenhum teste objetivo para determinar quão boa é uma solução, de modo que o teste envolve tão somente o julgamento subjetivo das várias partes envolvidas e interessadas;
- Não há nenhum conjunto pré-existente de soluções potenciais, nem um conjunto bem descrito das propriedades de tais soluções. Devem ser descobertas durante a análise de problemas, que nos levará à descoberta dos requisitos;
- Cada problema é suficientemente complexo, de forma que nenhum outro problema será idêntico a esse. Além disso, cada problema pode ser considerado o **sintoma** de um outro problema, o que significa que é difícil isolar o problema e também escolher um nível apropriado de abstração para descrevê-lo;
- Problemas complexos tendem a nos induzir a erros, o que muitas vezes têm profundas implicações políticas, éticas e/ou profissionais: as partes interessadas tendem a ser **intolerantes** em relação a erros; e isso a despeito de todos os esforços despendidos pela área de Recursos Humanos (RH) das empresas, e também das escolas de Administração e Empreendedorismo, que dizem ser melhor errar **logo**, pois assim aprendemos e podemos fazer as coisas melhores;
- O excesso de trabalho, quando um engenheiro de requisitos precisa chegar a uma declaração acordada do problema, **é o próprio problema**. Esse tipo de situação não apenas é mal definido por falta de experiência, mas também porque tende a desafiar a própria descrição.

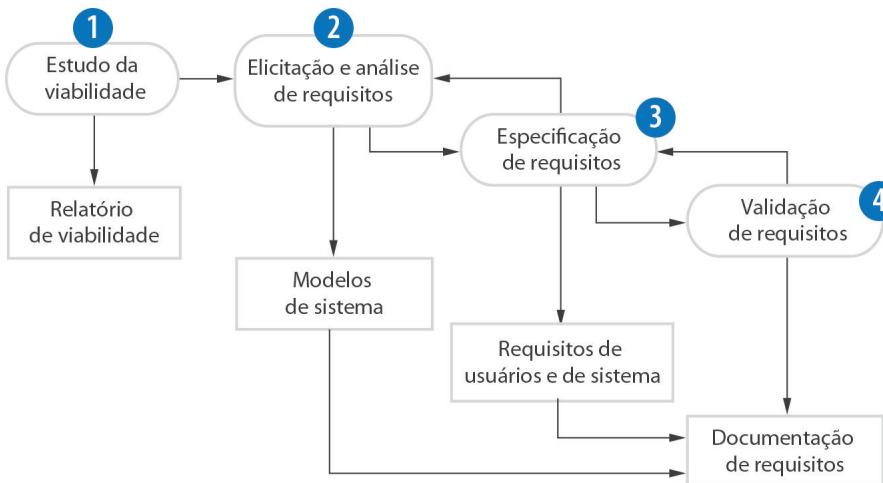


Figura 1

Fonte: Adaptado de SOMMERVILLE, 2011

Visão geral do processo da engenharia de requisitos e suas quatro fases:

- Estudo de viabilidade:** é feita uma estimativa acerca da possibilidade de se satisfazerem as necessidades do usuário identificado e usando as tecnologias atuais de *hardware* e *software*. Verifica se o sistema será rentável e se pode ser desenvolvido considerando as restrições orçamentárias;
- Elicitação e análise de requisitos:** observação e análise dos sistemas existentes, discussão com os potenciais usuários e compradores, análise das tarefas etc. Pode compreender o desenvolvimento de um protótipo;
- Especificação dos requisitos:** traduzir as informações obtidas na fase de análise em um documento que defina um conjunto de requisitos;
- Validação dos requisitos:** verifica os requisitos quanto ao realismo, à consistência e completude. Podem ser descobertos erros e o documento deve ser modificado para a correção desses problemas. A análise de requisitos é contínua, pois novos requisitos podem emergir durante o processo.

Requisitos do Sistema

Conforme o SEBoK – *System Engineering Body of Knowledge*, **Corpo de Conhecimentos em Engenharia de Sistemas** (2011 p. 1) –, os requisitos do sistema são todos os requisitos, no nível do sistema, que descrevem as funções que deve, como um todo, cumprir para satisfazer às necessidades e aos requisitos das partes interessadas. É expresso por uma combinação apropriada de declarações textuais, visualizações e requisitos não funcionais. Este último expressa os níveis de segurança, confiabilidade etc., que serão necessários.

Os requisitos do sistema são importantes porque formam a **base da arquitetura do sistema**. Atividades referentes à integração e verificação são referências para a validação e aceitação das partes interessadas, e são, por fim, um meio de comunicação entre os técnicos que precisam interagir durante todo o ciclo de vida do desenvolvimento do sistema.

Como você pôde notar, após a leitura e análise do relatório de necessidades – resultante de um processo termo de abertura de um projeto e de descrições iniciais –, a primeira parte importante é a engenharia de requisitos do *software*. Define-se por duas atividades, ilustradas por duas questões: primeiramente, quais soluções serão adotadas? Em segundo lugar, quais especificações serão escritas para que uma solução, que possa ser comprovada a partir de testes, seja compreensível, permitindo a manutenção e alongando o seu ciclo de vida com qualidade?

Análise de Problemas – Requisitos

A engenharia de requisitos oferece uma gama de técnicas para lidar com a **complexidade** dos problemas, tais como:

- **Abstração** envolve ignorar detalhes para que se possa ter uma visão geral e abrangente. Por exemplo, faz-se abstração sempre que se toma algum conjunto de atividades realizadas por seres humanos, descrevendo-o em termos de **sistema**;
- **Decomposição** envolve “quebrar” um conjunto de fenômenos em partes menores para que possamos analisá-las separadamente umas das outras. Tais processos de decomposição nunca são perfeitos devido ao **acoplamento** entre as partes, mas uma decomposição – o mesmo que dizer **análise** – bem-feita ainda nos oferece **insights** sobre como as coisas funcionam;
- **Projeção** significa adotar uma visão ou perspectiva específica que pertença a outrem, tentando imaginar e descrever – “projetar” – tudo aquilo que seria relevante para este ou aquele indivíduo, este ou aquele grupo de pessoas. Ao contrário do que ocorre na decomposição, as perspectivas **não** se destinam a serem – tratadas como – independentes, mas naturalmente atreladas umas às outras.

Assim, a primeira coisa à qual nos referimos aqui é a palavra **problema** que, no âmbito da engenharia de requisitos, é sinalizado pela própria palavra: **requisito**.

Requisitos só existem porque há alguém que quer alguma coisa – um produto, serviço, *software* –; esse alguém é um cliente e, portanto, em alguns casos/projetos, os requisitos são entendidos e descritos como uma lista de recursos ou funções, propriedades, restrições e tantas outras coisas exigidas pelo cliente.

Na prática, raramente há um único cliente, mas antes um **conjunto** diversificado de pessoas que serão afetadas, de alguma forma, pelo sistema. De modo geral, para a simplificação em projetos de *software*, entende-se por **cliente** a pessoa/área ou empresa que deseja o *software* e que pagará pelo qual; o usuário, por sua vez, é aquele que efetivamente **utilizará o software** desenvolvido a partir da lista de requisitos entregue pela engenharia de requisitos. Portanto, **quem o utilizará sabe muito bem do que precisa** – mas não menospreze ou subestime as expectativas e interesses de quem colocará a mão no bolso!

Todas essas pessoas e grupos de pessoas podem ter objetivos e interesses os mais variados e, muitas vezes, conflitantes. Podem não ser explícitos, ou ainda

difícies de articular e comunicar objetivamente. Podem **não saber o que querem**. Nestas circunstâncias, simplesmente perguntar **o que necessitam** não costuma ser muito frutífero, infelizmente.



Um requisito de *software* é uma descrição dos principais recursos de um produto de *software*, o seu fluxo de informações, comportamento e atributos. Em suma, um requisito de *software* fornece uma estrutura básica para o desenvolvimento de um produto de *software*. O grau de compreensibilidade, precisão e rigor da descrição fornecida por um documento de requisitos tende a ser diretamente proporcional ao grau de qualidade do produto final (PETERS; WITOLD, 2001, p. 102).

Os principais componentes – e os seus respectivos fatores – de uma análise de requisitos, com vista à excelência do resultado final, são os seguintes:

- **Funcional:** serve para identificar as atividades do sistema, as ações;
- **Comportamental:** demonstra a sequência e se há sobreposição de funções do sistema em uma hierarquia de atividades de controle; tal sequenciamento serve para perceber e controlar as funções do sistema em diversos patamares – são os controles;
- **Não comportamental:** toda a parte de engenharia de pessoas e garantia de qualidade; ações e atributos que influenciam na excelência do resultado final.

O que se espera, com tudo isto, é um **termo**, isto é, uma especificação de requisitos, uma descrição do sistema e um plano de garantia da qualidade, bem como dos atributos que precisam ser seguidos pelo time de desenvolvimento.



Importante!

A ISO 2007 define um requisito como uma declaração que identifica um produto ou processa uma característica ou restrição operacional, funcional ou de projeto, que é inequívoca, testável ou mensurável, e necessária para a aceitabilidade do produto ou processo.

Para que possamos identificar todas essas variáveis, há uma sequência de etapas bem conhecida da engenharia de requisitos:

1. **Concepção:** aqui são definidos o **escopo** e a **natureza** do sistema;
2. **Elicitação:** começa-se a **reunir** e **organizar** os requisitos para o *software*;
3. **Elaboração:** refinar os requisitos que foram reunidos e minimamente organizados;
4. **Negociação:** são determinadas as prioridades de cada, anotados os requisitos essenciais e – o que é mais importante – solucionados os conflitos entre os diferentes requisitos;
5. **Especificação:** agora, os requisitos são reunidos em um único produto, sendo o resultado da engenharia de requisitos;

6. Validação: estabelece-se a **qualidade** dos requisitos: se são inequívocos, consistentes, completos etc.;

7. Gerenciamento: por fim, são gerenciadas as **mudanças** que os requisitos devem sofrer ao longo do tempo de vida do projeto.

A análise dos requisitos ou dos problemas, como já nos referimos, serve para identificar o ambiente, os itens produzidos, as principais funções executadas pelas pessoas e os equipamentos utilizados para a produção de um produto, os seus métodos e o seu cronograma operacional.

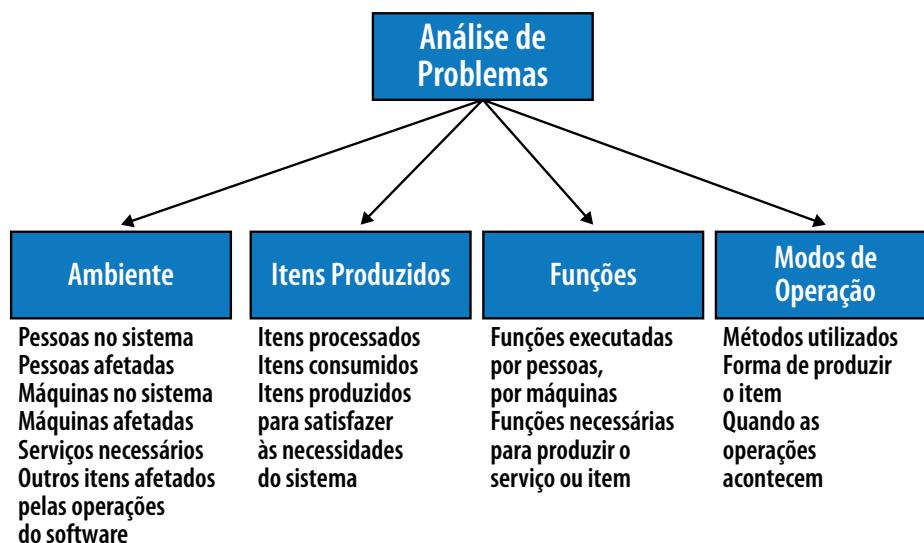


Figura 2 – Componentes que comumente são levados em consideração ao se fazer uma análise de problemas/requisitos

Classificação de Requisitos

Elaborar e definir **requisitos** é uma tarefa complexa que inclui uma série de processos, tais como **elicitação, análise, especificação, validação e gerenciamento**. É necessário saber como são classificados antes de conhecer os principais tipos de requisitos para produtos de *software*:

- **Requisitos de negócios:** incluem declarações de alto nível de objetivos e necessidades;
- **Requisitos das partes interessadas:** são as necessidades de determinados grupos que formam as partes interessadas discretas. É necessário especificá-los para definir o que cada parte interessada espera em termos de **solução** e **resultado final**;
- **Requisitos da solução:** as propriedades que um produto deve apresentar para atender da melhor maneira possível às necessidades das partes interessadas e do próprio negócio como um todo;

- » **Requisitos não funcionais:** são as características gerais de um sistema. São também conhecidos como **atributos de qualidade**;
- » Requisitos funcionais descrevem como um produto deve se comportar, os seus **recursos** e as suas **funções**. Ou seja, tudo aquilo que o faz funcionar como se espera que opere. Qualquer requisito funcional **não cumprido** impactará severamente o funcionamento, a ponto de fazer com que, no limite, simplesmente **não funcione**.
- **Requisitos de transição:** é um conjunto **adicional** de requisitos empregados em momentos de **transição** entre operações ou sistemas, definindo o que é necessário, da parte de uma organização, para conduzir-se com **êxito** de seu estado atual para o estado desejado a partir do novo produto/software.

Essa classificação de requisitos é a mais comumente utilizada, e você, como profissional de TI – Tecnologia da Informação –, conseguirá identificá-los com relativa facilidade. Todavia, o SEBoK, que é o acervo mais atualizado de engenharia de software, além de ser a fonte mais confiável, tornou-se um **padrão** em nossa indústria. Portanto, oferecemos para estudo a seguinte Tabela:

Tabela 1 – Exemplo de classificação de requisitos do sistema

Tipos de requisitos do sistema	Descrição
Requisitos funcionais	Descrevem qualitativamente as funções ou tarefas do sistema a serem executadas em operação.
Requisitos de desempenho	Definem quantitativamente a extensão, ou quão bem e sob quais condições uma função ou tarefa deve ser executada – por exemplo, taxas, velocidades etc. São requisitos quantitativos de desempenho do sistema e são individualmente verificáveis. Observe que pode haver mais de um requisito de desempenho associado a uma única função, requisito funcional ou tarefa.
Requisitos de <i>interface</i>	Definem de que modo o sistema se faz necessário para interagir ou trocar material, energia ou informações, com sistemas externos – <i>interface externa</i> –, ou como os elementos do sistema dentro do sistema, incluindo elementos humanos, interagem uns com os outros – <i>interface interna</i> . Os requisitos de <i>interface</i> incluem conexões físicas – <i>interfaces físicas</i> – com sistemas externos ou elementos internos do sistema que suportam interações ou trocas.
Requisitos operacionais	Definem as condições operacionais ou propriedades necessárias para o sistema operar ou existir. Esse tipo de requisito inclui fatores humanos, ergonomia, disponibilidade, facilidade de manutenção, confiabilidade e segurança.
Requisitos de modos e/ou estados	Definem os vários modos operacionais do sistema e os eventos que conduzem a transições de modos.
Restrições físicas	Definem as restrições de peso, volume e dimensão aplicáveis aos elementos do sistema.
Restrições de design	Definem os limites das opções disponíveis para o projetista de uma solução, impondo limites imutáveis: por exemplo, o sistema deve incorporar um legado ou elemento do sistema fornecido, ou determinados dados devem ser mantidos em um repositório <i>on-line</i> ?

Tipos de requisitos do sistema	Descrição
Condições ambientais	Definem as condições ambientais a serem encontradas pelo sistema em seus diferentes modos operacionais. Isto deve abordar o ambiente natural – por exemplo, o vento, a chuva, temperatura, fauna, o sal, a poeira, radiação etc. –, efeitos ambientais induzidos ou autoinduzidos – por exemplo, o movimento, choque, ruído, eletromagnetismo, térmico etc. –, e ameaças ao ambiente social – por exemplo, as legais, políticas, econômicas, sociais, comerciais etc.
Requisitos logísticos	Definem as condições logísticas necessárias para a utilização contínua do sistema. Esses requisitos incluem a manutenção – o fornecimento de instalações, suporte de nível, pessoal de suporte, as peças sobressalentes, o treinamento, a documentação técnica etc. –, a embalagem, o manuseio e transporte.
Políticas e regulamentos	Definem políticas organizacionais relevantes ou aplicáveis, ou requisitos regulatórios que possam afetar a operação ou o desempenho do sistema – por exemplo, políticas trabalhistas, relatórios para agência regulatória, critérios de saúde ou segurança etc.
Restrições de custo e cronograma	Definem, por exemplo, o custo de um único exemplar do sistema, a data de entrega esperada do primeiro exemplar etc.

Como vimos, os **requisitos de solução** são importantes, pois dizem respeito diretamente ao produto de *software* – sendo importante aprofundar-se neste ponto.

Requisitos Funcionais

Os requisitos funcionais são as **operações** desejadas e esperadas de um programa ou sistema, conforme definido no desenvolvimento de *software* e na engenharia de sistemas. Descrevem a função **final** desejada de um sistema operando conforme os parâmetros normais, de modo a assegurar que o projeto seja adequado para gerar o produto almejado, assim como o produto final atinja o seu potencial de projeto para atender às expectativas do usuário.

Comumente, um requisito funcional é uma funcionalidade básica ou um comportamento desejado, documentado de forma clara e quantitativa. Os requisitos funcionais, no que se refere a *softwares*, são complementados por requisitos de **qualidade** e requisitos **técnicos**, denominados **requisitos não funcionais**.

Segundo Waslawick (2014), os requisitos funcionais podem ser identificados como:

- **Evidentes:** funções executadas com o conhecimento do usuário. Esses requisitos geralmente correspondem à troca de informações entre o usuário e sistema, tais como consultas e entradas de dados que fluem pela interface do sistema;
- **Ocultos:** funções desempenhadas pelo sistema **sem** o conhecimento explícito do usuário. Geralmente estas funções são operações matemáticas e atualizações de dados, realizadas ocultamente pelo sistema, sem conhecimento explícito do usuário – mas como consequência de outras funções desempenhadas pelo usuário. Requisitos funcionais ocultos são executados **internamente** pelo sistema. Assim, embora não apareçam explicitamente como casos de uso, devem estar adequadamente associados aos quais para serem lembrados no momento do

projeto e da implementação. Neste caso, o usuário não solicita explicitamente que o sistema execute determinada operação. Como se trata de atividade executada automaticamente pelo sistema, é por princípio um requisito **oculto**.



Importante!

Quando um produto falha em requisitos funcionais, geralmente significa que o produto é de baixa qualidade e possivelmente inútil.

Alguns dos requisitos funcionais mais típicos incluem – mas não se limitam a:

- Regras do negócio;
- Correções de transação, ajustes e cancelamentos;
- Funções administrativas;
- Autenticações – usuário e senha;
- Níveis de autorização – alcada;
- Acompanhamento de auditoria – *logs*;
- *Interfaces* externas;
- Requisitos de certificação;
- Requisitos de relatório;
- Data histórica;
- Requisitos legais ou regulamentares.

Os requisitos funcionais geralmente são descritos **em texto**, ainda que se trate de um projeto tradicional ou ágil. No entanto, podem ser também visuais, apresentados no formato de:

- **Documento de especificação de requisitos de software:** contém descrições de funções e recursos que o produto deve fornecer. O documento também define restrições e suposições. Inclusive pode ser um único documento que comunica requisitos funcionais, ou pode acompanhar outras documentações de *software*, tais como histórias de usuários e casos de uso, nesse caso em metodologia ágil;
- **Casos de uso:** trata-se de um artefato comportamental da *UML – Unified Modeling Language* – que descreve a interação entre o sistema e os usuários, levando à obtenção de metas/objetivos específicos;
- **Histórias de usuários:** é uma descrição documentada de um recurso de *software visto da perspectiva do usuário final*. Portanto, deve descrever o que exatamente o usuário deseja do sistema para que este o atenda como esperado. Em projetos ágeis, as histórias de usuários são organizadas em um *backlog* – coisas para fazer: uma lista ordenada de funções do produto. Atualmente, as histórias de usuários são consideradas o melhor formato para itens de *backlog*;

- **Estrutura de decomposição do trabalho, ou EAP** – Estrutura Analítica do Projeto –, ou seja, uma decomposição **funcional**. É um documento visual que ilustra de que maneira os processos complexos se dividem em suas componentes mais simples. EAP é uma abordagem eficaz e que permite uma análise independente de cada parte, além de ajudar a capturar a imagem completa do projeto;
- **Protótipos:** em se tratando de *software*, podemos entender este quesito de forma abrangente, para diferentes tipos de entregas em estágio inicial, criadas para mostrar como os requisitos devem ser implementados. Os protótipos ajudam a dirimir as lacunas de visão e permitem que as partes interessadas e equipes clarifiquem áreas complicadas de produtos em vias de desenvolvimento. Tradicionalmente, os protótipos apresentam como a solução funcionará, fornecendo exemplos de como os usuários interagirão em suas tarefas. Comumente os clientes pedem que o protótipo seja submetido à grande quantidade de usuários para que possam ver como a camada de apresentação e distribuição de elementos aparecem na tela do computador. Com isso busca-se melhorar a **usabilidade**, ordenação de campos e/ou arquitetura da informação;
- **Modelos e diagramas:** quaisquer outros artefatos gráficos que possam ser úteis para a visualização da proposta de solução, desde que definidos como padrão e que sejam aceitos tanto interna como externamente – fábricas de *software*, de teste de *software*, escritórios de gerenciamento de projetos e todos os demais.

Como exemplo de requisito funcional entre produto/*software*, especificamente para uma embalagem, pode ser a capacidade de conter fluido sem vazamento.

Ademais, podem ser requisitos funcionais para um *software* os seguintes aspectos:

- O acesso ao sistema deverá ser realizado mediante identificador de usuário e senha pessoal e intransferível;
- O usuário deve ser capaz de pesquisar qualquer conjunto inicial de bancos de dados, ou selecionar um subconjunto dentro desse;
- O sistema deve fornecer visualizadores apropriados para que o usuário leia os documentos na área de armazenamento do documento;
- A cada pedido deve ser alocado um identificador único que o usuário deve ser capaz de copiar para a área de armazenamento permanente da conta.

Requisitos Não Funcionais

Os requisitos não funcionais descrevem como um sistema deve se comportar e estabelece restrições para a sua funcionalidade – **requisito funcional**. Esse tipo de requisito também é conhecido como **atributo de qualidade do sistema**. É essencial para garantir a usabilidade e eficácia de todo sistema de *software*. Falhar em atender aos requisitos não funcionais pode resultar em sistemas que não satisfazem às necessidades do usuário. Ademais, permite **impor restrições** no sistema.

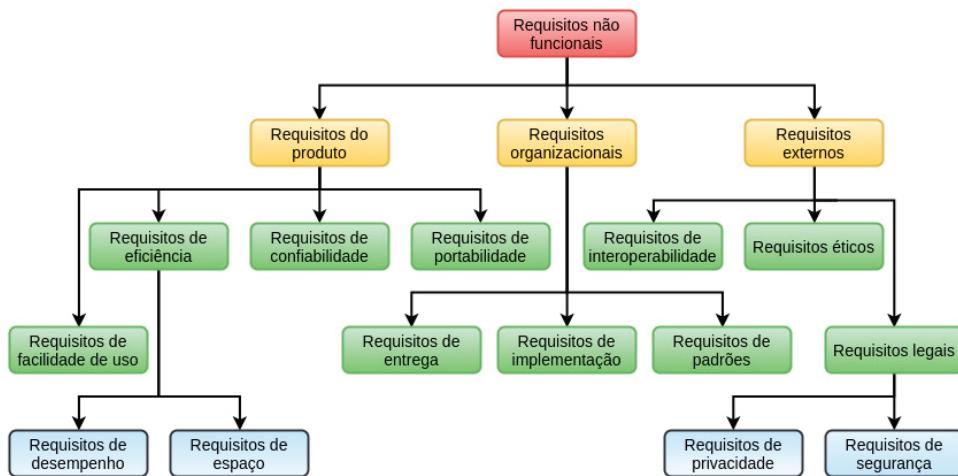


Figura 3 – Tipos de requisitos não funcionais

Fonte: Wikimedia Commons/github.com

De acordo com Antonio Mendes da Silva Filho (2008, p. 8-12), para requisitos não funcionais associados a produtos há os seguintes elementos:

Usabilidade: é um dos atributos de qualidade ou requisitos não funcionais de qualquer sistema interativo, ou seja, no qual ocorre interação entre o sistema e seres humanos. A noção de usabilidade vem do fato de que qualquer sistema projetado para ser utilizado pelas pessoas deveria ser fácil de aprender e fácil de usar, tornando, assim, fácil e agradável a realização de qualquer tarefa. Requisitos de usabilidade especificam tanto o nível de desempenho quanto a satisfação do usuário no uso do sistema;

Manutenibilidade: é geralmente empregado quando nos referimos às modificações feitas após o sistema de *software* ter sido disponibilizado para uso. Na realidade, o termo **manutenibilidade** é um tanto abrangente já que ele envolve tanto a atividade de reparo (de algum defeito existente no sistema de *software*), quanto a atividade de alteração/evolução de características existentes ou adição de novas funcionalidades não previstas ou capturadas no projeto inicial;

Confiabilidade: é a probabilidade de o *software* não causar uma falha num sistema durante um determinado período de tempo sob condições especificadas. A probabilidade é uma função da existência de defeitos no *software*. Assim, os estímulos recebidos por um sistema determinam a existência ou não de algum defeito. Em outras palavras, a confiabilidade de *software*, geralmente definida em termos de comportamento estatístico, é a probabilidade de que o *software* irá operar como desejado num intervalo de tempo conhecido. Também, a confiabilidade caracteriza-se um atributo de qualidade de *software* que implica que um sistema executará suas funções como esperado;

Disponibilidade: é uma medida de quão disponível o sistema estaria para uso, isto é, quão disponível o sistema estaria para efetuar um serviço solicitado por algum usuário. Por exemplo, um serviço de um sistema de *software* terá uma disponibilidade de 999/1.000. Isto significa que dentre um conjunto

de 1.000 solicitações de serviço, 999 deverão ser atendidas. Esta métrica é muito importante em sistemas de telecomunicações, por exemplo.

- **Taxa de ocorrência de falha:** é uma medida da frequência na qual o sistema falha em prover um serviço como esperado pelo usuário, ou seja, a frequência na qual um comportamento inesperado é provável de ser observado. Por exemplo, se temos uma taxa de ocorrência de falha de 2/1.000, isto significa que 2 falhas são prováveis de acontecerem para cada 1.000 unidades de tempo;
- **Probabilidade de falha durante fase operacional:** é uma medida da probabilidade que o sistema irá comportar-se de maneira inesperada quando em operação. Esta métrica é de suma importância em sistemas críticos que requerem uma operação contínua;
- **Tempo médio até a ocorrência de falha, ou Mean Time to Failure (MTTF):** é uma medida do tempo entre falhas observadas. Note que esta métrica oferece um indicativo de quanto tempo o sistema permanecerá operacional antes que uma falha aconteça.

Desempenho: é um atributo de qualidade importante para sistemas de *software*. Considere, por exemplo, um sistema de uma administradora de cartões de crédito. Em tal sistema, um projetista ou engenheiro de *software* poderia considerar os requisitos de desempenho para obter uma resposta de tempo para autorização de compras por cartão. Note que os requisitos de desempenho têm impacto mais global sobre o sistema e, por essa razão, estão entre os requisitos não funcionais mais importantes.

- **Requisitos de resposta:** especificam o tempo de resposta de um sistema de *software* aceitável para usuários. Neste caso, um projetista poderia especificar que o sistema deveria responder à solicitação de um serviço específico de um usuário dentro de um intervalo de 2 segundos;
- **Requisitos de processamento (*throughput*):** estes requisitos especificam a quantidade de dados que deveria ser processada num determinado período de tempo. Um exemplo seria exigir que o sistema de *software* possa processar, no mínimo, 6 transações por segundo;
- **Requisitos de temporização:** este tipo de requisito especifica quanto rapidamente o sistema deveria coletar dados de entrada de sensores antes que outras leituras de dados de entrada, feitas posteriormente, sobrescrevam os dados anteriores;
- **Requisitos de espaço:** em alguns casos, os requisitos de espaço podem ser considerados. Aqui, podemos nos referir à memória principal ou secundária. Por exemplo, a memória principal para executar uma aplicação poderia ser considerada como um requisito de desempenho uma vez que ela está relacionada ao comportamento do sistema em tempo de execução.

Portabilidade: pode ser definida como a facilidade na qual o *software* pode ser transferido de um sistema computacional ou ambiente para outro. Em outras palavras, o *software* é dito portável se ele pode ser executado em ambientes distintos. Note que o termo ambiente pode referir-se tanto

à plataforma de *hardware* quanto a um ambiente de *software* como, por exemplo, um sistema operacional específico;

Reusabilidade: uma característica das engenharias é fazer uso de projetos existentes a fim de reutilizar componentes já desenvolvidos, objetivando minimizar o esforço em novos projetos. Dessa forma, componentes que já tenham sido desenvolvidos e testados podem ser reutilizados. Considere os elevados níveis de reusabilidade que encontramos tanto na indústria de automóveis quanto de aparelhos eletrônicos. Na indústria de automóveis, por exemplo, um motor é geralmente reutilizado de um modelo de carro para outro;

Segurança: em um sistema de *software*, este requisito não funcional caracteriza a segurança de que acessos não autorizados ao sistema e dados associados não serão permitidos. Portanto, é assegurada a integridade do sistema quanto a ataques intencionais ou acidentes. Dessa forma, a segurança é vista como a probabilidade de que a ameaça de algum tipo será repelida.

A seguir, exemplos de requisitos não funcionais genéricos:

- Os usuários devem alterar a senha de *login* original imediatamente após o primeiro *login*. Além disso, a senha inicial nunca deve ser reutilizada;
- Os funcionários não podem atualizar as próprias informações de salário. Tal tentativa deve ser relatada ao administrador de segurança;
- Toda tentativa malsucedida de um usuário para acessar um item de dados deve ser registrada em uma trilha de auditoria;
- Um *site* deve ter capacidade suficiente para lidar com 20 milhões de usuários sem que isso afete o seu desempenho;
- O *software* deve ser portável. Assim, mudar de um sistema operacional para outro não criará nenhum problema;
- A privacidade da informação, a exportação de tecnologias restritas, os direitos de propriedade intelectual etc., tudo isto deve ser auditado.

Vejamos exemplos de requisitos não funcionais de produto (SOUZA, 2017):

Requisitos de produtos:

Tabela 2 – Requisitos de usabilidade

RNF01	O sistema deverá ser operável por usuários sem a necessidade de treinamento prévio.
RNF02	O sistema deverá fazer uso de <i>design</i> responsivo na implementação de suas <i>interfaces</i> gráficas, de modo a se comportar adequadamente em navegadores acessados via computador, <i>smartphone</i> e <i>tablet</i> .
RNF03	O sistema deverá ser disponibilizado em português.

Tabela 3 – Requisitos de confiabilidade

RNF04	Apenas usuários-administradores devem possuir acesso à camada de administração do sistema.
RNF05	A senha do usuário será gravada/trafegada utilizando-se o algoritmo PBKDF2 com uma hash SHA256.
RNF06	As únicas informações que o sistema poderá exibir, no tocante a cartões de crédito salvos previamente na conta do usuário, são os últimos 4 dígitos, a data de validade e operadora.

Tabela 4 – Requisitos de portabilidade

RNF07	O sistema deverá funcionar nos navegadores <i>Google Chrome</i> , <i>Mozilla Firefox</i> e <i>Safari</i> .
-------	------------------------------------------------------------------------------------------------------------

Requisitos organizacionais:

Tabela 5 – Requisitos de entrega

RNF08	O desenvolvimento do sistema deverá ser feito de maneira incremental com os PI seguindo a fundamentação do SAFe.
RNF09	Os entregáveis relacionados à arquitetura do sistema e ao nível gerencial de equipe deverão ser elaborados de acordo com as datas estimadas no <i>Kanban</i> do projeto.
RNF10	A priorização das <i>features</i> para cada entrega de incremento do produto deverá ser feita com a estimativa e os critérios de aceitação recomendados pelo SAFe.

Tabela 6 – Requisitos de implementação

RNF11	O sistema será desenvolvido utilizando a linguagem <i>Python</i> sob o framework <i>Django</i> .
RNF12	Todas as variáveis de entrada terão “vazio”, ou equivalentes, como valor <i>default</i> .

Tabela 7 – Requisitos padrões

RNF13	O sistema será desenvolvido utilizando o paradigma de programação orientado a objetos.
-------	----------------------------------------------------------------------------------------

Requisitos externos:

Tabela 8 – Requisitos éticos

RNF14	O sistema deverá se comunicar com o banco de dados <i>SQLite 3</i> .
RNF15	O sistema não apresentará aos usuários quaisquer dados de caráter privativo, tais como informações pessoais de outros usuários.

Tabela 9 – Requisitos legais

RNF16	O sistema deverá atender às normas legais no que se refere à comercialização <i>on-line</i> de produtos.
-------	----------------------------------------------------------------------------------------------------------

Partes e Questionamentos em uma Especificação de Requisitos de Software

Quando escrevemos uma especificação de requisitos de software, comumente lidamos com cinco questões básicas, listadas na Figura a seguir:

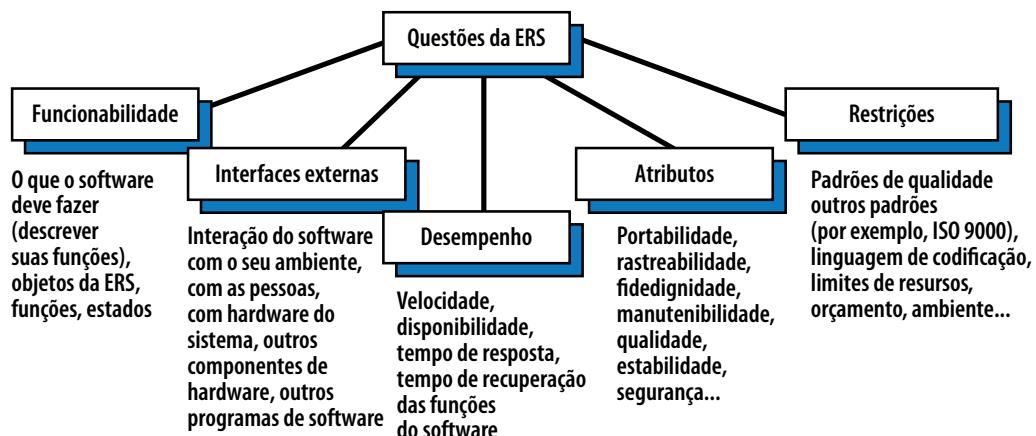


Figura 4 – Questões básicas consideradas ao se escrever uma especificação de requisitos

É importante termos em mente que os requisitos geram um documento que deve ser mantido e atualizado, pois, independentemente do paradigma de engenharia que utilizemos – como, por exemplo, cascata, espiral, que são tradicionais, ou ágeis, tais como *Scrum*, *XP*, *FDD* ou *TDD* –, devemos gerar a documentação e administrá-la, uma vez que os requisitos sofrem mudanças, atualizações, versionamentos, acréscimos e exclusões. A seguir você verá as partes de uma especificação através de seu índice – esta estrutura em questão foi adaptada do DoD USA – Departamento de Defesa dos Estados Unidos da América:



Figura 5 – Partes de uma especificação de requisitos de software

Especificar os requisitos é tão fundamental para o sucesso de um projeto de *software* que é feito um acompanhamento dos maiores problemas relatados em termos de **insucesso**. Eis o que relaciona Pfleeger (2004, p. 112):

- Requisitos incompletos com 13,1%;
- Falta de envolvimento por parte do usuário com 12,4%;
- Falta de recursos com 10,6%;
- Expectativas não realistas com 9,9%;
- Falta de apoio dos executivos com 9,3%;
- Modificações nos requisitos e nas especificações com 8,7%;
- Falta de planejamento com 8,1%;
- O sistema não era mais necessário com 7,5%.

Neste momento, é importante refletir sobre esses dados e trabalhar com as possibilidades **mais destruidoras** de uma boa especificação de requisitos, que exigirá de você muito trabalho e treino para superá-las.

- Falta de conhecimento do negócio pelo analista;
- Uso inadequado do pouco tempo que possui para fazer a especificação;
- Falta de método e organização da informação coletada;
- Subestimar a inteligência e o conhecimento do usuário e cliente;
- Inexperiência associada à prepotência;
- Falta de intimidade com ferramentas de apoio à especificação de requisitos.

Material Complementar

Indicações para saber mais sobre os assuntos abordados nesta Unidade:

▶ Vídeos

O que é requisito

<https://youtu.be/oo06hyLuFNU>

Requisitos funcionais e não funcionais

<https://youtu.be/hsAtfdfk5N4>

📄 Leitura

A (suma) importância da engenharia de requisitos

<http://bit.ly/38j396l>

Processo de engenharia de requisitos aplicado a requisitos não funcionais de desempenho – um estudo de caso

<http://bit.ly/2Pseclc>

Referências

- DA SILVA FILHO, A. M. **Engenharia de software 3** – requisitos não funcionais. 2008. Disponível em: <<https://www.devmedia.com.br/artigo-engenharia-de-software-3-requisitos-nao-funcionais/9525>>. Acesso em: 19 jul. 2019.
- ISO-SEI 2007. *Requirements management process area and requirements development process area*. In: **Capability Maturity Model Integrated (CMMI) for development**, version 1.2. Pittsburgh, PA, USA: SEI/CMU, 2007.
- PETERS, J. F.; WITOLD, P. **Engenharia de software**. 3. ed. São Paulo: Campus, 2001.
- SEBoK. **Guide to the System Engineering Body of Knowledge**. 2011. Disponível em: <[https://www.sebokwiki.org/wiki/Guide_to_the_Systems_Engineering_Body_of_Knowledge_\(SEBoK\)](https://www.sebokwiki.org/wiki/Guide_to_the_Systems_Engineering_Body_of_Knowledge_(SEBoK))>. Acesso em: 19 jul. 2019.
- SOMMERVILLE, I. **Engenharia de software**. 8. ed. [S.l.: s.n.], 2011.
- SOUZA, R. F. **Requisitos não funcionais**. Github, 2017. Disponível em: <https://github.com/Desenho-2-2017/Ecom_merci/wiki/Requisitos-n%C3%A3o-Funcionais>. Acesso em: 19 jul. 2019.
- WAZLAWICK, R. S. **Análise e projeto orientado a objetos para sistemas de informação** – modelando com UML, OCL e IFML. New York: Elsevier, 2014.



Cruzeiro do Sul
Educacional