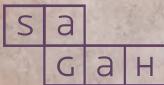


ENGENHARIA DE REQUISITOS

Sheila Reinehr



SOLUÇÕES
EDUCACIONAIS
INTEGRADAS

Requisitos de *software*

Objetivos de aprendizagem

Ao final deste texto, você deve apresentar os seguintes aprendizados:

- Classificar os requisitos de um projeto de *software*.
- Priorizar os requisitos em um projeto de *software*.
- Aplicar critérios de qualidade para avaliar a qualidade dos requisitos de *software*.

Introdução

Produtos de *software* fazem parte do nosso dia a dia em praticamente todas as atividades que realizamos, desde as mais simples até as mais sofisticadas. Empresas de todos os setores também dependem fortemente de produtos de *software*, que são utilizados tanto para a realização de atividades produtivas quanto para as de controle e gestão. Quando um *software* falha, seus transtornos são sentidos imediatamente e impactam na forma como as atividades cotidianas de empresas e indivíduos são realizadas.

Uma das etapas do ciclo de desenvolvimento de *software* que tem o maior potencial de causar danos futuros ao projeto e ao produto final em si é a engenharia de requisitos. Quando não compreendemos adequadamente o que deve ser construído, falhamos para estimar prazos e recursos, gerando prejuízos financeiros para o projeto. Mais importante do que isso, uma falha na compreensão dos requisitos pode se propagar para o restante das atividades do desenvolvimento, gerando produtos que não satisfazem as necessidades e que podem, inclusive, gerar danos físicos ou levar à perda de vidas.

Sabemos, no entanto, que os recursos para o desenvolvimento não são infinitos e a velocidade do mercado exige que novos produtos ou funcionalidades sejam lançados cada vez mais rápido. Então como fazer para equilibrar o tempo disponível para o desenvolvimento com a qualidade exigida do produto final? O primeiro passo é garantir bons requisitos, de modo que a agilidade possa ser acompanhada de qualidade.

Neste capítulo você vai estudar os diferentes tipos de requisito e quais os desdobramentos dessas categorias. Verá também como os requisitos podem ser priorizados em função de suas características e das características do contexto. Por fim, vai ler sobre como aplicar critérios para analisar a qualidade dos requisitos, para que possam contribuir de forma efetiva para a qualidade das próximas etapas do desenvolvimento de *software*.

1 Tipos de requisitos de *software*

Inicialmente, precisamos definir o que é requisito. Inúmeras definições podem ser encontradas e optamos, neste livro, por aderir ao *Glossário padrão de terminologia de engenharia de software* do *Institute of Electrical and Electronic Engineers* (IEEE, 1990) que define requisito como:

1. Uma condição ou capacidade necessária para um usuário resolver um problema ou alcançar um objetivo.
2. Uma condição ou capacidade que deve ser atendida ou tida por um sistema ou componente do sistema para satisfazer a um contrato, padrão, especificação ou outro documento formalmente imposto.
3. Uma representação documentada de uma condição ou capacidade conforme estabelecido em 1 e 2.

Ao olhar mais detidamente os termos envolvidos nessa definição, você percebe que os principais aspectos dos requisitos estão cobertos: foco no problema que a solução deve resolver; foco na satisfação a uma imposição (que pode, por exemplo, ser uma legislação); e, finalmente, foco na representação documentada. Note ainda que essa definição não direciona para um método específico de desenvolvimento, uma vez que não especifica “como”, mas sim “o quê”.

Outra forma de entender o que é requisito é se apoiar na definição de Sommerville e Sawyer (1997), que estabelecem que requisitos são a especificação do que precisa ser implementado, são descrições de como o sistema deve se comportar, ou são uma propriedade ou atributo do sistema. Os requisitos podem, por exemplo, ser uma restrição no processo de desenvolvimento do sistema. Aqui temos o conceito de requisito do processo, que será tratado posteriormente neste capítulo.

O termo **requisito** tem diversas interpretações diferentes na engenharia de *software*. Cada autor apresenta uma forma diferente de classificar os requisitos. Pohl e Rupp (2015) classificam os requisitos de acordo com as definições do Quadro 1.

Quadro 1. Tipos de requisitos de acordo com Pohl e Rupp (2015)

Tipo do requisito	Definição
Requisito funcional	É um requisito que se refere a um resultado de comportamento que deve ser provido por uma funcionalidade do sistema.
Requisito de qualidade	É um requisito que pertence a uma preocupação com a qualidade que não é coberta pelos requisitos funcionais.
Restrição	É um requisito que limita o espaço da solução além do que é necessário para atender aos requisitos funcionais e requisitos de qualidade.

Fonte: Adaptado de Pohl e Rupp (2015).

Um **requisito funcional** é, como o próprio nome diz, uma funcionalidade requerida pelos *stakeholders* para cumprir algum objetivo de negócio. Representa o que os desenvolvedores deverão implementar para que os usuários possam realizar as suas atividades. Geralmente, os requisitos funcionais são expressos em frases do tipo “o sistema deve”. Por exemplo: “O sistema deve permitir que o usuário pague com cartão de débito ou crédito”.

Os **requisitos de qualidade** se referem a como o *software* vai operar sob determinadas circunstâncias e, geralmente, estão relacionados a questões como desempenho, disponibilidade, usabilidade, portabilidade, escalabilidade etc. Eles também são chamados de **requisitos não funcionais**.

É comum que os requisitos não funcionais sejam negligenciados no início do projeto — e aí reside um grande problema, pois eles, geralmente, têm forte impacto sobre a arquitetura da aplicação, e são os requisitos mais difíceis de se retificar posteriormente. Para conseguirmos atender a um requisito específico de desempenho, por exemplo, é necessário fazer escolhas arquiteturais que darão suporte a esse requisito. Se isso não for planejado em tempo de projeto (*design*), será muito difícil corrigir depois que a implementação estiver avançada.

As **restrições** são consideradas como os requisitos sobre os quais a equipe de desenvolvimento não tem gestão. Eles não são implementados no produto de *software*, mas influenciam a forma como o produto será implementado. Geralmente, relacionam-se com as escolhas sobre tecnologias (*hardware*, ambientes, linguagens de programação etc.) e processos de desenvolvimento (ciclo de vida, métodos, procedimentos de gestão etc.), além de restrições do próprio projeto que gera o produto (prazos, custos, recursos etc.). As restrições são também comumente chamadas de **requisitos de processo** e **requisitos de projeto**.

Wiegers e Beatty (2013) utilizam uma visão um pouco diferente, incluindo, além dos tipos em si, o nível de abstração dos requisitos, conforme você pode ver no Quadro 2.

Quadro 2. Tipos de requisitos de acordo com Wiegers e Beatty (2013)

Termo	Definição
Requisito de negócio	Um objetivo de alto nível de negócios da organização que constrói um produto ou de um cliente que o adquire.
Regra de negócio	Uma política, um guia, um padrão, uma regulamentação que define ou restringe algum aspecto de negócios. Não é um requisito do <i>software</i> em si, mas a origem de diversos tipos de requisitos de <i>software</i> .
Restrição	Uma restrição que é imposta sobre as escolhas disponíveis para o desenvolvedor para o projeto (<i>design</i>) e construção de um produto.
Requisito de interface externa	Uma descrição de uma conexão entre um sistema de <i>software</i> e um usuário, outro sistema de <i>software</i> ou um dispositivo de <i>hardware</i> .
Característica (<i>feature</i>)	Uma ou mais capacidades do sistema logicamente relacionadas que fornecem valor para um usuário e são descritas por um conjunto de requisitos funcionais.
Requisito funcional	Uma descrição de um comportamento que o sistema deve exibir sob condições específicas.
Requisito não funcional	Uma descrição de uma propriedade ou característica que o sistema deve exibir ou uma restrição que ele deve respeitar.

(Continua)

(Continuação)

Quadro 2. Tipos de requisitos de acordo com Wieggers e Beatty (2013)

Termo	Definição
Atributo de qualidade	Um tipo de requisito não funcional que descreve um serviço ou característica de desempenho de um produto.
Requisito de sistema	Um requisito de alto nível para um produto que contém múltiplos subsistemas, que pode ser todo de <i>software</i> ou de <i>software</i> e <i>hardware</i> .
Requisito de usuário	Um objetivo ou tarefa que classes específicas de usuários devem ser capazes de executar com um sistema ou um atributo de produto desejado.

Fonte: Adaptado de Wieggers e Beatty (2013).

De acordo com a visão de Wieggers e Beatty (2013), os **requisitos de negócio** estão mais diretamente ligados aos motivos pelos quais um *software* está sendo desenvolvido, ou seja, quais são os objetivos de negócios que se deseja atingir e qual o valor que esse produto agregará ao negócio. Geralmente, são providos pelo patrocinador, executivos da organização ou pelo pessoal de produto (gerência de *marketing*, por exemplo). Os requisitos de negócio são como grandes objetivos que todos precisam estar de acordo, pois serão a base da maioria das decisões futuras sobre o projeto e até mesmo a priorização das entregas. Com base nos requisitos de negócio também serão avaliadas as solicitações de mudanças que certamente surgirão ao longo do desenvolvimento.

Embora pareça bastante óbvio que um projeto deva ser iniciado de forma alinhada aos objetivos de negócio, nem sempre isso acontece, pois, muitas vezes, a visão sobre esse relacionamento não é muito clara. Como consequência, a equipe técnica pode ter que lidar com visões conflitantes entre os diversos *stakeholders*, o que pode ter impacto nas atividades de desenvolvimento e no resultado final. Outro efeito colateral possível é que a equipe tente abranger as visões de todos os possíveis *stakeholders*, deixando de focar naqueles que são mais relevantes, levando o produto a abranger um escopo muito maior do que o inicialmente previsto. Discutiremos mais esse tópico nas próximas seções.



Exemplo

Exemplos de benefícios quantificáveis financeiramente sob a ótica do negócio:

- incrementar as vendas em 25% em 6 meses;
- aumentar a quantidade de engajamentos na página da marca em 15% nos próximos 3 meses;
- reduzir os gastos com vendedores presenciais em 40% até o final do próximo ano.

Exemplos de benefícios não financeiros podem incluir:

- aumentar o índice de satisfação dos clientes em 10% no próximo trimestre;
- receber no máximo 10 chamados por mês no service desk após 3 meses da implantação do produto.

Como se pode observar nos quadros anteriores, existem formas diferentes de classificar os requisitos, mas, basicamente, podemos considerar que existem requisitos que são do **produto** de *software* em si, ou seja, que serão construídos pela equipe de desenvolvimento e farão parte do produto entregue; e existem requisitos que não são do produto, ou seja, fazem parte do **projeto** ou do **processo**. Os requisitos que são do produto, podem ser **funcionais** (o que o *software* deve fazer) ou **não funcionais** (como o *software* deve fazer). Os requisitos de projeto e de processo são sempre não funcionais.

Outro conceito importante é o de **requisito de sistema**. Embora, na maioria das empresas, o termo **sistema** seja usado para designar um produto de *software*, seu conceito é mais amplo do que isso. Entende-se por sistema, a composição de *hardware*, *software* e processos. Portanto, requisitos de sistema são requisitos de mais alto nível, que devem ser atendidos pelo produto como um todo, e o requisito de *software* se refere ao requisito do sistema que é alocado ao componente de *software*. Isso se torna particularmente importante quando falamos de sistemas ciberfísicos (CPS, *cyber-physical systems*), cuja composição envolve a parte física (ambiente, sensores, atuadores) e a parte ciber (*software*). Esse é o caso, por exemplo, das funcionalidades de *park assist* (estacionamento automático) nos automóveis de luxo. Um requisito de sistema poderia ser: “O sistema deve buscar uma vaga para estacionar o veículo”. Esse requisito do sistema envolve elementos físicos (sensores para detectar espaços vazios) e elementos de *software* (para identificar se o carro cabe naquele espaço).



Saiba mais

As normas da ISO oferecem um bom suporte para todos os processos relacionados a requisitos:

- Para mais detalhes sobre processos relacionados a requisitos de sistema, uma boa dica é olhar a norma internacional ISO/IEC 15288:2015, *Systems and Software Engineering — Systems Lyfe Cycle Processes*. Nela é possível identificar o processo de definição de requisitos de sistema, que estabelece as atividades e os resultados esperados quando se executa o processo.
- Para mais detalhes sobre processos relacionados aos requisitos de *software*, você pode usar a norma internacional ISO/IEC/IEEE 12207:2017, *Systems and Software Engineering — Software Lyfe Cycle Processes*. Procure pelos processos de definição de requisitos de sistema/*software*. Embora essa norma seja dedicada aos processos de *software*, ela trata os requisitos de forma mais ampla, incluindo requisitos de sistema.
- Para mais detalhes sobre requisitos de forma mais ampla, consulte a norma internacional ISO/IEC 29148:2018, *Systems and Software Engineering — Lyfe Cycle Processes — Requirements Engineering*. Ela inclui mais detalhes sobre como implementar os processos que estão descritos nas duas normas anteriores.

É importante compreendermos adequadamente esses diferentes tipos de requisitos, pois cada um deles será desdobrado em um artefato diferente do ciclo de vida. Os requisitos relacionados ao **produto** farão parte de um documento de especificação de requisitos (ou artefato similar, como cartões com histórias de usuário nos métodos ágeis), enquanto os requisitos de **processo** e de **projeto** serão parte do plano de projeto (ou artefato que tenha função similar). Já requisitos de **sistema** se desdobrarão em requisitos relacionados à parte física (*hardware*) e requisitos relacionados à parte de *software*, que podem estar em um mesmo artefato, mas que se desdobrarão em implementações distintas.

2 Priorização de requisitos

Uma das maiores causas de fracassos em projetos está relacionada ao seu tamanho. Quanto maior o projeto, maiores são as chances de ele não atingir suas metas de prazo, custo, escopo e qualidade. Uma forma de minimizar esses efeitos é dividi-lo em partes menores e mais facilmente gerenciáveis. Esse é o princípio que foi adotado pelos modelos de ciclo de vida iterativos e incrementais e, mais recentemente, pelas metodologias ágeis, que preveem a divisão de trabalho em *sprints* ou iterações, com durações curtas, de,

no máximo, um mês. Dessa forma, os requisitos podem ser conhecidos logo no início, mas eles serão detalhados à medida que forem sendo priorizados para a alocação em uma entrega.

A forma de priorizar os requisitos vai depender do contexto em que o projeto está inserido e das próprias características do produto e da tecnologia envolvida. Alguns dos fatores que podem ser usados na priorização envolvem, mas não estão restritos a:

- **Potencial de valor a ser agregado ao negócio pelo requisito:** requisitos que agregam mais valor ao negócio geralmente são alocados às primeiras entregas, enquanto requisitos de menor valor são alocados a versões posteriores.
- **Dependência do requisito em relação a outros requisitos:** muitas vezes um requisito não pode ser implementado isoladamente, requerendo que sejam priorizados outros requisitos que dão suporte a ele, na mesma versão do produto.
- **Análise de requisitos implícitos que podem estar invisíveis:** no momento da priorização de requisitos, pode ser que requisitos implícitos que estavam invisíveis em um primeiro momento sejam identificados e precisem ser priorizados juntamente com o requisito principal. Isso é muito comum, por exemplo, nos requisitos de segurança de acesso, que podem implicar em uma série de requisitos de apoio (que podem até mesmo virar um subsistema).
- **Experiência com a tecnologia por parte da equipe do projeto:** a falta de experiência da equipe de desenvolvimento com a tecnologia pode levar à priorização de requisitos que não sejam exatamente os que geram mais valor ao cliente, mas sim aqueles que possibilitam o aprendizado mais rápido e com menor risco para o projeto como um todo.
- **Experiência no domínio de aplicação por parte da equipe:** analogamente ao anterior, a falta de experiência da equipe com o domínio de aplicação requer cuidados redobrados na priorização, especialmente, quando houver o risco de a equipe subestimar a complexidade de um requisito em função de sua falta de experiência com o domínio de negócio. Nesses casos, a proximidade com o usuário, cliente ou seus representantes é fundamental.
- **Relacionamentos com outros sistemas (*hardware ou software*):** quando o *software* implicar em relacionamentos com outros *softwares* ou com *hardware*, na forma de um sistema, é necessário que as dependências

entre os requisitos sejam claramente explicitadas, de modo que o desenvolvimento de requisitos dependentes possa ser realizado de forma síncrona e não isolada.

- **Demandas de implementações para atender a determinações de ordem legal ou regulatória:** é comum que demandas de ordem legal ou que sejam demandadas por órgãos regulamentadores tenham datas específicas para entrar em vigor. Requisitos que implementam essas necessidades precisam ser priorizados em versões compatíveis com as datas demandadas.
- **Requisitos que não possuem alternativas manuais aceitáveis:** há casos em que o procedimento manual para a realização da atividade de negócio é oneroso e sujeito a falhas. Nesses casos, requisitos que substituem esses processos são priorizados em detrimento de outros, cujos procedimentos manuais sejam menos onerosos ou representem menos riscos ao negócio.

A priorização pode ser feita por um único representante do usuário, como um *product owner* (PO) do método Scrum, ou por um grupo de *stakeholders*. Isso pode acontecer virtualmente, usando ferramentas de colaboração, ou em *workshops* de priorização presenciais. A forma vai depender da quantidade de *stakeholders* envolvidos e do nível de conflito que possa haver entre os requisitos de produto a serem priorizados e as restrições do projeto/processo. Quanto mais críticos forem esses conflitos, maior a necessidade de se utilizar procedimentos colaborativos de decisão.

3 Critérios de qualidade de um bom requisito

Não importa o modelo de ciclo de vida ou o processo de desenvolvimento utilizado, um requisito mal compreendido e mal documentado será sempre um problema para o projeto que envolve *software*. Por esse motivo, é imprescindível que os requisitos, independentemente de sua categoria, estejam claramente identificados, de modo que as atividades que dependem deles possam ser realizadas com sucesso. Como podemos estimar adequadamente o esforço para a implementação de um requisito se sua descrição está ambígua e imprecisa? Como evitar o retrabalho no desenvolvimento se o requisito não foi detalhado de forma suficiente? Como prever os testes necessários se não houver detalhes para isso?

Primeiramente, é importante não confundir um requisito bem especificado com uma etapa de requisitos infinita, que se perde nos mínimos detalhes e que não avança na entrega de valor para o cliente. Aqui é ressaltada a importância de termos o requisito especificado de forma clara e em nível de detalhe compatível com o contexto. Por **contexto** entendemos as variáveis que podem afetar essa decisão, como experiência da equipe de desenvolvimento com a tecnologia e com a área de negócio, criticidade do produto em desenvolvimento, impacto dos possíveis erros que possam ser derivados (*software* de missão crítica, por exemplo) e nível de envolvimento dos *stakeholders*.



Fique atento

É comum achar que, ao utilizar métodos ágeis, não é necessário gastar muito tempo com os requisitos, pois eles vão, invariavelmente, mudar. Na verdade, temos sim que dispensar todo o esforço necessário para termos certeza de que compreendemos adequadamente cada demanda que vai para desenvolvimento, do contrário contribuiremos, entre outras coisas, para o aumento da insatisfação do cliente com o produto entregue.

O que acontece em métodos ágeis como o Scrum é que o *product backlog* tem itens em níveis diferentes de maturidade. Os itens que já estão sendo priorizados para serem produzidos na próxima *sprint* devem estar maduros o suficiente para serem estimados, alocados à *sprint* e implementados. Haverá, por outro lado, itens menos maduros no *product backlog*, que serão detalhados no momento oportuno, conforme a sua prioridade.

De acordo com Schwaber e Sutherland (2018), itens de alta prioridade no *product backlog* costumam ser mais claros e detalhados do que itens de prioridade mais baixa. Estimativas mais precisas são feitas com base na maior clareza e no maior nível de detalhe; quanto menos prioritário, menos detalhado. Note que os criadores do Scrum ressaltam que os itens menos prioritários estarão menos detalhados, mas no momento em que se tornarem prioritários, terão que estar claros e detalhados.

De acordo com a Norma Internacional ISO/IEC/IEEE 29148 (ISO/IEC/IEEE, 2018), um requisito bem especificado contém um ou mais destes itens:

- deve ser atendido ou tido por um sistema para resolver um problema, atingir um objetivo ou tratar de uma preocupação de um *stakeholder*;
- é qualificado por condições mensuráveis;

- é limitado por restrições;
- define o desempenho do sistema quando usado por um *stakeholder* específico ou é a capacidade correspondente de um sistema, mas não a capacidade de um usuário, operador ou outro *stakeholder*;
- pode ser verificado (isto é, a realização de um requisito no sistema pode ser demonstrada).

Como documentar um requisito

Diversas são as formas de documentar um requisito, incluindo linguagem natural, tabelas, planilhas, diagramas, vídeos, áudios, fotos etc. Neste capítulo, vamos nos concentrar na forma escrita, usando a linguagem natural. Ela pode ser utilizada tanto para os requisitos funcionais quanto para os não funcionais, sejam eles de produto, sejam de projeto ou processo. Serve também para especificar os requisitos ou objetivos de negócios, que vão nortear todo o desenvolvimento. As demais formas podem funcionar como um apoio ou complemento ao que está descrito em linguagem natural.

A primeira preocupação é que a linguagem natural apresenta ambiguidades que podem comprometer a precisão dos requisitos e a sua compreensão pelo público-alvo. Escrever requisitos não é como escrever qualquer outro tipo de texto, de ficção ou não. Não há uma forma de ensinar a escrever um bom requisito, pois isso vem com a experiência, mas, algumas dicas podem ajudar a começar:

- **Ótica da escrita:** lembre-se que um requisito tem que ser visto sob a ótica de quem lê o documento, e não sob a ótica de quem escreve. Muitas vezes o requisito está completamente claro para quem escreveu, mas gera dúvidas para quem lê. Nesse caso, ele precisa ser revisado.
- **Tamanho das frases:** frases longas dificultam a leitura e podem levar a interpretações incorretas. Dê preferência a frases curtas e diretas. Releia algumas vezes e elimine qualquer palavra que não agregue compreensão ao conteúdo.
- **Correção na escrita:** os requisitos constituem um documento técnico e devem, portanto, seguir as normas da língua portuguesa escrita. Isso quer dizer que devem estar corretos gramaticalmente e sem erros de digitação. Inclua aí uma preocupação extra com a pontuação (que pode modificar completamente o sentido da frase).

- **Uso da voz ativa:** dê preferência para frases usando a voz ativa (por exemplo, “o sistema deve prover uma função de cálculo do imposto”) em vez da voz passiva (por exemplo “uma função de cálculo do imposto deve ser provida”). Misturar voz passiva e ativa também torna o texto difícil de ler.
- **Uso da forma positiva:** também é preferível usar a forma positiva e evitar a negativa do tipo “não deve”. Alguns chamam essa forma de requisito inverso, mas ela deve ser evitada. Frases com muitas negativas também geram complexidade no entendimento. Tudo o que precisamos ler duas vezes para entender não está bem escrito.
- **Termos ambíguos:** há termos que, naturalmente, levam a ambiguidades e indeterminações, e devem ser removidos de qualquer especificação de requisitos, como alguns, muitos, poucos, melhor, pior, robusto, adequado, rápido, amigável. Também há expressões que dificultam a precisão, como quando apropriado, quando possível, quando aplicável, se possível, se necessário.
- **Tempo verbal:** o tempo verbal empregado pode trazer um significado adicional implícito ao requisito, e isso nem sempre é desejável. Por exemplo: “o sistema deve” sugere que é um requisito obrigatório, mas o “sistema deveria” sugere uma interpretação de que o sistema poderia não fazer, implicando em um requisito menos prioritário. Questões de prioridade não devem ser tratadas pelo tempo verbal, mas sim por atributos de prioridade específicos, uma vez que podem mudar ao longo do tempo.
- **Acrônimos e jargões:** acrônimos (siglas) e jargões também constituem um ponto frágil na especificação de requisitos. O recomendado é manter um glossário compartilhado entre todos os membros do projeto (internos e externos), de modo que não sejam usados termos de forma inconsistente.
- **O que e não como:** lembre-se de que, idealmente, os requisitos devem definir o que deve ser feito, e não como deve ser implementado.



Saiba mais

As normas internacionais da ISO (International Organization for Standardization), no Brasil editadas pela ABNT (Associação Brasileira de Normas Técnicas), fazem uma clara distinção entre os itens que são escritos com o tempo verbal **deve** (*shall*) e com o tempo verbal **deveria** (*should*). As cláusulas de uma norma que possuem o verbo **deve** são obrigatórias e, no caso de normas certificadoras, serão requeridas pelo auditor como parte do processo de auditoria para concessão do certificado. Já as cláusulas que estão descritas com o verbo **deveria** não são exigidas, e a empresa pode opcionalmente não implementá-las (ISO/IEC/IEEE, 2018).

Uma forma para a escrita dos requisitos é fazê-la em três etapas:

1. Escrever a primeira versão.
2. Realizar a leitura crítica dessa versão se colocando no lugar do leitor.
3. Solicitar a revisão de um colega, a chamada **revisão por pares**.

Com o tempo, a escrita vai se tornando mais natural e os primeiros erros não serão mais cometidos, especialmente se eles, anteriormente, levaram a alguma consequência indesejada, como o retrabalho de implementação por ambiguidade nos requisitos. Leve a sério os *feedbacks* recebidos dos seus pares e elabore seu próprio *checklist* para autorrevisão dos requisitos — isso vai ajudá-lo a aprimorar essa habilidade ao longo do tempo.

Características de um bom requisito

Alguns critérios de qualidade que podem ser utilizados para nortear a escrita e a verificação dos requisitos estão listados a seguir, baseados nas recomendações de Wiegers e Beatty (2013) e da Norma Internacional ISO/IEC/IEEE 29148 (ISO/IEC/IEEE, 2018). Eles podem ser organizados no formato de um *checklist*, a ser aplicado para analisar cada requisito antes de sua liberação para a equipe de implementação, ou mesmo antes de liberá-los para a revisão por pares. O Quadro 3 apresenta as características de um bom requisito.

Quadro 3. Características de um bom requisito

Atributo	Definição
Completo	O requisito está especificado de forma completa e possibilita que o desenvolvedor o implemente.
Correto	O requisito reflete o que o usuário, cliente ou seus representantes desejam.
Único	O requisito descreve uma única capacidade, característica, restrição ou atributo de qualidade.
Viável	O requisito é viável técnica e financeiramente para ser implementado, de acordo com as restrições do projeto.
Necessário	O requisito tem um motivo de existir, que é representado pelo seu relacionamento com uma fonte de informação e com um objetivo de negócio.
Priorizado	O requisito tem uma prioridade atribuída para que possa ser alocado a uma versão do <i>software</i> .
Não ambíguo	O requisito não contém ambiguidades que levem os <i>stakeholders</i> a interpretá-lo de forma diferente.
Verificável	O requisito pode ser verificado posteriormente à sua implementação.
Conforme	O requisito está em conformidade com os padrões de especificação estabelecidos, se houver.

Fonte: Adaptado de Wiegers e Beatty (2013) e ISO/IEC/IEEE (2018).

Características de um bom conjunto de requisitos

Como você pode observar no quadro anterior, cada requisito, individualmente, deveria atender às características listadas para ser considerado um bom requisito, apto a servir de base para as atividades seguintes do ciclo de desenvolvimento. Adicionalmente, existem características que precisam ser consideradas para o conjunto de requisitos de modo que a solução possa entregar o valor esperado pelos *stakeholders*, respeitando as suas restrições. Entende-se por **solução** não necessariamente o produto completo, mas uma versão entregue do produto.

No Quadro 4, compilado a partir das definições de Wiegers e Beatty (2013) e da Norma Internacional ISO/IEC/IEEE 29148 (ISO/IEC/IEEE, 2018), podemos observar as características de um bom conjunto de requisitos. Diferentemente do quadro anterior, vemos que aqui aparecem atributos que são analisados para um ou mais requisitos. Por exemplo, analisar se o conjunto de requisitos está completo se refere a buscar pelos requisitos implícitos ou invisíveis, por exemplo. Pode ser que determinado requisito só possa ser implementado com a preparação de algum tipo de infraestrutura, como a existência de cadastros prévios, por exemplo. Muitas vezes, esses cadastros não são explicitamente identificados pelos *stakeholders*, por acharem que já estavam subentendidos.

Outro aspecto importante é a viabilidade de um requisito, quando analisado em conjunto com os demais. Podemos ter, por exemplo, um requisito de desempenho (tempo de resposta) que seja incompatível com um requisito de processo (utilizar os equipamentos disponíveis atualmente).

Quadro 4. Características de um bom conjunto de requisitos

Atributo	Definição
Completo	O conjunto de requisitos está completo.
Consistente	Os requisitos são consistentes entre si e com os requisitos de mais alto nível dos quais são originários, não apresentando conflitos.
Modificável	Cada requisito tem uma identificação única que permite que ele seja modificado quando necessário.
Compreensível	O conjunto de requisitos está escrito de forma que é possível identificar claramente o que é esperado do <i>software</i> no ambiente do qual ele faz parte.
Viável	O conjunto de requisitos pode ser executado dentro das restrições estabelecidas (técnicas, de custo e prazo), dentro de riscos aceitáveis.
Capaz de ser validado	O conjunto de requisitos pode ser validado quanto às necessidades dos <i>stakeholders</i> dentro das restrições (como custo, prazo, técnica, conformidade com regulamentações e legislações).
Rastreável	O conjunto de requisitos pode ser rastreado às suas origens (<i>backward</i>) e também aos demais elementos, como requisitos derivados, elementos de <i>design</i> , código e casos de teste (<i>forward</i>).

Fonte: Adaptado de Wiegers e Beatty (2013) e ISO/IEC/IEEE (2018).

Nível de detalhamento dos requisitos

Uma pergunta difícil de responder é sobre quão detalhados devem ser os requisitos. Isso vai depender de alguns fatores relacionados ao produto de *software* em si e ao contexto em que ele está sendo desenvolvido, conforme você pode acompanhar a seguir (WIEGERS; BEATLY, 2013).

■ Mais detalhes:

- o trabalho está sendo executado para um cliente externo;
- o desenvolvimento e/ou o teste serão terceirizados;
- os membros do projeto estão geograficamente dispersos;
- os testes de sistema serão realizados com base nos requisitos;
- estimativas precisas são necessárias;
- a rastreabilidade entre os requisitos é necessária.

■ Menos detalhes:

- o trabalho está sendo realizado internamente para a sua empresa;
- os clientes estão bastante envolvidos;
- os desenvolvedores possuem uma grande experiência no domínio;
- existem precedentes, como no caso em que uma aplicação está sendo substituída;
- um pacote de solução será utilizado.

Em resumo, se a equipe é experiente no domínio do negócio e os clientes estão próximos e acessíveis, o nível de detalhamento dos requisitos pode ser menor, pois os riscos envolvidos também serão menores. No entanto, quando a equipe não é tão experiente no domínio do negócio, os riscos são maiores e, portanto, um nível maior de detalhamento dos requisitos é necessário.



Referências

IEEE. *IEEE Std 610.12-1990: IEEE Standard Glossary of Software Engineering Terminology*. Los Alamitos: IEEE Computer Society Press, 1990.

ISO/IEC/IEEE. *ISO/IEC/IEEE 12207:2017: systems and software engineering: software life cycle processes*. Switzerland: ISO, 2017.

ISO/IEC/IEEE. *ISO/IEC/IEEE 15288:2015: systems and software engineering: systems life cycle processes*. Switzerland: ISO, 2015.

ISO/IEC/IEEE. *ISO/IEC/IEEE 29148:2018: systems and software engineering: life cycle processes: requirements engineering*. Switzerland: ISO, 2018.

POHL, K.; RUPP, C. *Requirements engineering fundamentals*: a study guide for the certified professional for requirements engineering exam, foundation level, IREB Compliant. 2. ed. Santa Barbara: Rock Nook, 2015.

SCHWABER, K.; SUTHERLAND, J. *Guia do SCRUM*: um guia definitivo para o SCRUM: as regras do jogo. [S. l.], 2018. Disponível em: <https://www.scrumguides.org/index.html>. Acesso em: 30 dez. 2019.

SOMMERRVILLE, I.; SAWYER, P. *Requirements engineering*: a good practice guide. Chichester: John Wiley & Sons, 1997.

WIEGERS, K. E.; BEATTY, J. *Software requirements*. 3. ed. Redmond: Microsoft Press, 2013.

Leituras recomendadas

BOURQUE, P.; FAIRLEY, R. (ed.). *SWEBOk*: guide to the software engineering body of knowledge version 3. Washington: IEEE Computer Society, 2014.

PRESSMAN, R.; MAXIM, B. *Engenharia de software*: uma abordagem profissional. 8. ed. Porto Alegre: AMGH, 2016.

WIEGERS, K. *More about software requirements*: thorny issues and practical advices. Redmond: Microsoft Press, 2006.

Conteúdo:

