



Martin Glinz

Hans van Loenhoud

Stephen Steel

palco stan

Manual para o CPRE Foundation Level de acordo com o padrão IREB

Educação e Treinamento para

Profissional Certificado em Engenharia de Requisitos (CPRE)

Nível da Fundação

Versão 1.1.0

setembro de 2022

Termos de uso

Todo o conteúdo deste documento, especialmente textos, fotografias, gráficos, diagramas, tabelas, definições e modelos, são protegidos por direitos autorais. Copyright © 2022 deste manual é de responsabilidade dos autores. Todos os (co-)autores deste documento transferiram o direito exclusivo de uso para IREB eV

Qualquer uso do manual ou de seus componentes, em particular a cópia, distribuição (publicação), tradução ou reprodução, requer o consentimento prévio do IREB eV

Qualquer pessoa tem o direito de usar o conteúdo do manual dentro do escopo dos atos de uso permitidos pela lei de direitos autorais, em particular para citá-los corretamente de acordo com as regras acadêmicas reconhecidas.

As instituições de ensino têm o direito de usar o conteúdo do manual para fins de ensino sob referência correta ao trabalho.

A utilização para fins publicitários só é permitida com o consentimento prévio da IREB eV

Reconhecimentos

O conteúdo deste manual foi revisado por Rainer Grau, Karol Frühauf e Camille Salinesi. Tracey Duffy fez uma revisão em inglês. Stan Bühne e Stefan Sturm fizeram a edição final.

A versão 1.0.0 foi aprovada para lançamento em 11 de novembro de 2020 pelo Conselho do IREB por recomendação de Xavier Franch e Frank Houdek.

A versão 1.1.0 foi aprovada para lançamento em 15 de agosto de 2022 pelo IREB ExCo.

Agradecemos a todos pelo envolvimento.

Índice

Índice.....	3
Prefácio.....	7
Compreendendo as caixas de texto neste manual	8
Histórico da versão	9
1 Introdução e Visão geral.....	10
1.1 Engenharia de Requisitos: O que	10
1.2 Engenharia de Requisitos: Por quê.....	11
1.3 Engenharia de Requisitos: Onde.....	12
1.4 Engenharia de Requisitos: Como.....	13
1.5 O Papel e as Tarefas de um Engenheiro de Requisitos.....	13
1.6 O que Aprender sobre Engenharia de Requisitos	14
1.7 Leitura Adicional	14
2 Princípios Fundamentais da Engenharia de Requisitos.....	16
2.1 Visão Geral dos Princípios	16
2.2 Os Princípios Explicados	16
2.2.1 Princípio 1 – Orientação de valor: Os requisitos são um meio para um fim, não um fim em si.....	16
2.2.2 Princípio 2 – Stakeholders: ER é sobre satisfazer os desejos e necessidades dos stakeholders.....	18
2.2.3 Princípio 3 – Compreensão compartilhada: O desenvolvimento de sistemas bem-sucedidos é impossível sem uma base comum	19
2.2.4 Princípio 4 – Contexto: Os sistemas não podem ser entendidos isoladamente.....	21
2.2.5 Princípio 5 – Problema, requisito, solução: Um triplô inevitavelmente entrelaçado.....	23
2.2.6 Princípio 6 – Validação: Requisitos não validados são inúteis	24
2.2.7 Princípio 7 – Evolução: A mudança de requisitos não é um acidente, mas o caso normal.....	25
2.2.8 Princípio 8 – Inovação: Mais do mesmo não é suficiente.....	26
2.2.9 Princípio 9 – Trabalho sistemático e disciplinado: Não podemos prescindir no RE	26
2.3 Leitura Adicional	27
3 Produtos de Trabalho e Práticas de Documentação	28
3.1 Produtos de Trabalho em Engenharia de Requisitos.....	28

3.1.1	Características dos Produtos de Trabalho.....	28
3.1.2	Níveis de Abstração.....	30
3.1.3	Nível de detalhe.....	31
3.1.4	Aspectos a Considerar	31
3.1.5	Diretrizes Gerais de Documentação	34
3.1.6	Planejamento do Produto de Trabalho.....	34
3.2	Produtos de Trabalho Baseados em Linguagem Natural.....	35
3.3	Produtos de Trabalho Baseados em Modelos	37
3.3.1	Modelos de Frases	37
3.3.2	Modelos de Formulário	39
3.3.3	Modelos de Documentos	40
3.3.4	Vantagens e Desvantagens	41
3.4	Produtos de Trabalho Baseados em Modelo.....	41
3.4.1	O Papel dos Modelos na Engenharia de Requisitos	42
3.4.2	Contexto do Sistema de Modelagem.....	48
3.4.3	Estrutura de Modelagem e Dados.....	52
3.4.4	Função e Fluxo de Modelagem	54
3.4.5	Modelando Estado e Comportamento.....	57
3.4.6	Modelos complementares	59
3.5	Glossários.....	62
3.6	Documentos de Requisitos e Estruturas de Documentação	63
3.7	Protótipos em Engenharia de Requisitos.....	64
3.8	Critérios de Qualidade para Produtos de Trabalho e Requisitos	65
3.9	Leitura Adicional	66
4	Práticas para Elaboração de Requisitos.....	68
4.1	Fontes para Requisitos	69
4.1.1	Partes interessadas	71
4.1.2	Documentos.....	76
4.1.3	Outros Sistemas	77
4.2	Elicitação de Requisitos	77
4.2.1	O Modelo Kano.....	79
4.2.2	Técnicas de coleta.....	82
4.2.3	Técnicas de Design e Geração de Idéias.....	85
4.3	Resolvendo Conflitos em Relação a Requisitos	89

4.3.1 Como você resolve um conflito de requisitos?.....	90
4.3.2 Tipos de Conflito.....	92
4.3.3 Técnicas de Resolução de Conflitos	94
4.4 Validação de Requisitos.....	96
4.4.1 Aspectos Importantes para Validação.....	97
4.4.2 Técnicas de Validação.....	98
4.5 Leitura Adicional	103
5 Processo e estrutura de trabalho.....	104
5.1 Fatores de influência	104
5.2 Facetas do Processo de Engenharia de Requisitos.....	106
5.2.1 Faceta do Tempo: Linear versus Iterativo	107
5.2.2 Faceta do Propósito: Prescritiva versus Explorativa.....	107
5.2.3 Faceta Alvo: Específico ao Cliente versus Orientado para o Mercado.....	108
5.2.4 Dicas e Advertências	109
5.2.5 Considerações Adicionais	110
5.3 Configurando um Processo de Engenharia de Requisitos	110
5.3.1 Combinações Típicas de Facetas.....	110
5.3.2 Outros Processos de RE.....	113
5.3.3 Como Configurar Processos de RE.....	113
5.4 Leitura Adicional	114
6 Práticas de Gerenciamento de Requisitos	115
6.1 O que é Gerenciamento de Requisitos?	116
6.2 Gerenciamento do ciclo de vida	117
6.3 Controle de versão.....	118
6.4 Configurações e linhas de base	120
6.5 Atributos e visualizações.....	122
6.6 Rastreabilidade	124
6.7 Manipulando Mudanças.....	126
6.8 Priorização	128
6.9 Leitura Adicional	130
7 Suporte de ferramenta.....	131
7.1 Ferramentas em Engenharia de Requisitos	131
7.2 Apresentando as ferramentas	133

7.2.1 Considerar todos os custos do ciclo de vida além dos custos de licença.....	133
7.2.2 Considere os Recursos Necessários	133
7.2.3 Evite Riscos Executando Projetos Pilotos.....	134
7.2.4 Avalie a ferramenta de acordo com os critérios definidos.....	134
7.2.5 Instruir os Funcionários sobre o Uso da Ferramenta	135
7.3 Leitura Adicional	135
8 Referências	136

Prefácio

Este manual fornece uma introdução à Engenharia de Requisitos com base no syllabus versão 3.0 para o Profissional Certificado para Engenharia de Requisitos (CPRE)—Nível Foundation de acordo com o padrão IREB. Complementa o programa e dirige-se a três grupos de leitores:

- *Alunos e profissionais* que desejam aprender sobre Engenharia de Requisitos e fazer o exame de certificação podem usar este manual como um livro complementar para cursos de treinamento oferecidos por provedores de treinamento, bem como para autoestudo e preparação individual para o exame de certificação. Este manual também pode ser usado para atualizar o conhecimento existente sobre Engenharia de Requisitos, por exemplo, ao se preparar para um curso e exame CPRE Advanced Level.
- *Os provedores de treinamento* que oferecem treinamentos no nível CPRE Foundation podem usar este manual como um complemento ao programa para desenvolver seus materiais de treinamento ou como um texto de estudo para os participantes em seus treinamentos.
- *Os profissionais* da indústria que desejam aplicar conceitos e conhecimentos comprovados de ER em seu trabalho prático encontrarão neste manual uma riqueza de informações úteis.

Este manual também fornece um link entre o plano de estudos, que lista e explica os objetivos de aprendizado, e a literatura sobre Engenharia de Requisitos. Cada capítulo vem com referências à literatura e dicas para leitura adicional. A estrutura do manual corresponde à estrutura do programa.

A terminologia usada neste manual é baseada no Glossário CPRE de Terminologia de Engenharia de Requisitos [Glin2020]. Recomendamos baixar este glossário do site do IREB e usá-lo como referência de terminologia.

Você encontra mais informações sobre o programa de certificação CPRE, incluindo o programa de estudos, glossário, regulamentos de exame e exemplos de perguntas do exame no site da IREB em <https://www.ireb.org>.

Tanto os autores quanto o IREB investiram tempo e esforço significativos na preparação, revisão e publicação deste manual. Esperamos que você goste de estudar este manual. Se você detectar algum erro ou tiver sugestões de melhoria, entre em contato conosco pelo e-mail info@ireb.org.

Gostaríamos de agradecer a todas as pessoas que contribuíram para a criação e publicação deste manual. Karol Frühauf, Rainer Grau e Camille Salinesi revisaram cuidadosamente o manuscrito e forneceram sugestões valiosas para melhorias. Tracey Duffy fez uma crítica em inglês. Também agradecemos aos Pastores do Conselho do IREB deste manual, Xavier Franch e Frank Houdek, por seus comentários e apoio. Stefan Sturm forneceu incentivo e apoio logístico. Agradecemos também a nossos cônjuges e familiares por sua paciência e apoio.

Martin Glinz, Hans van Loenhoud, Stefan Staal e Stan Bühne
novembro de 2020

Compreendendo as caixas de texto neste manual

O manual inclui quatro caixas de texto de cores diferentes que complementam o texto explicativo.

Estes são:

Definições (correspondentes ao Glossário [Glin2020])

dicas

Exemplos

Expressões

Histórico da versão

Data da versão		Comente	Autores
1.0.0	11 de novembro de 2020	Primeiro lançamento	Martin Glinz Hans van Loenhoud Stephen Steel palco stan
1.0.1	Dezembro 2020	Pequena atualização	
1.1.0	1º de setembro de 2022	Atualizações editoriais e alinhamento com o CPRE Foundation Level Syllabus atualizado v 3.1.0.	Martin Glinz Hans van Loenhoud Stephen Steel palco stan Wim Decoutre

1 Introdução e visão geral

Neste capítulo, você aprenderá sobre o que é a Engenharia de Requisitos (RE) e o valor que a RE traz.

1.1 Engenharia de Requisitos: O que

Desde o início da evolução humana, os seres humanos vêm construindo sistemas técnicos e organizacionais para *apoia*-los na conclusão de tarefas ou no alcance de objetivos.

Com o surgimento da engenharia, os humanos também começaram a construir sistemas que *automatizam* tarefas humanas.

Sempre que os humanos decidem construir um sistema para suportar ou automatizar tarefas humanas, eles precisam descobrir *o que construir*. Isso significa que eles precisam aprender sobre os desejos e necessidades das pessoas ou organizações que usarão o sistema, se beneficiarão dele ou serão impactados por ele. Em outras palavras, eles precisam saber sobre os *requisitos* desse sistema. Os requisitos formam a base para qualquer desenvolvimento ou evolução de sistemas ou partes deles. Os requisitos sempre existem, mesmo quando não são explicitamente capturados e documentados.

A necessidade de requisitos

O termo *requisito* denota três conceitos [Glin2020]:

Definição 1.1. Requisito: 1. Uma necessidade percebida por uma *parte interessada*. 2. Uma capacidade ou propriedade que um *sistema* deve ter. 3. Uma representação documentada de uma necessidade, capacidade ou propriedade.

Requerimento

Uma coleção de requisitos sistematicamente representada - normalmente para um sistema - que satisfaz determinados critérios é chamada de *especificação de requisitos*.

Distinguimos entre três tipos de requisitos:

tipos de requisitos

- Os *requisitos funcionais* dizem respeito a um resultado ou comportamento que deve ser fornecido por uma função de um sistema. Isso inclui requisitos de dados ou a interação de um sistema com seu ambiente.
- Os *requisitos de qualidade* referem-se a preocupações de qualidade que não são cobertas por requisitos funcionais — por exemplo, desempenho, disponibilidade, segurança ou confiabilidade.
- As *restrições* são requisitos que limitam o espaço da solução além do necessário para atender aos requisitos funcionais e requisitos de qualidade fornecidos.

Observe que lidar com requisitos para projetos ou processos de desenvolvimento está fora do escopo deste manual.

A distinção entre requisitos funcionais, requisitos de qualidade e restrições nem sempre é direta. Uma maneira comprovada de diferenciá-los é perguntar sobre a *preocupação* que um requisito aborda: se a preocupação for sobre os resultados, comportamento ou interações exigidos, temos um requisito funcional. Se for uma preocupação de qualidade que não esteja coberta pelos requisitos funcionais, temos um requisito de qualidade. Se a preocupação é restringir o espaço da solução, mas não é um requisito funcional nem de qualidade, temos uma restrição. A regra popular “*O que* o sistema deve fazer é requisito funcional versus *como* o sistema deve fazer é requisito de qualidade” freqüentemente leva a erros de classificação, particularmente quando os requisitos são especificados em grande detalhe ou quando os requisitos de qualidade são muito importantes.

Como distinguir entre requisitos funcionais, requisitos de qualidade e restrições

Por exemplo, o requisito “A ficha de cadastro do cliente deve conter campos para nome e nome do cliente, ocupando até 32 caracteres por campo, exibindo no mínimo 24 caracteres, à esquerda, com 12 pt. sanserif font” é um requisito funcional, embora contenha muitas informações sobre *como*. Como outro exemplo, considere um sistema que processa os dados de medição produzidos pelo detector de um acelerador de partículas de alta energia. Esses detectores produzem enormes quantidades de dados em tempo real. Se você perguntar a um físico “O que o sistema deve fazer?”, uma das primeiras respostas provavelmente seria que o sistema deve ser capaz de lidar com o volume de dados produzidos. No entanto, os requisitos relativos ao volume de dados ou à velocidade de processamento são requisitos de qualidade [Glin2007] e não requisitos funcionais.

Quando as pessoas adotam uma abordagem sistemática e disciplinada para a especificação e gerenciamento de requisitos, chamamos isso de *Engenharia de Requisitos (ER)*. A seguinte definição de Engenharia de Requisitos também reflete por que executamos RE.

Definição 1.2. Engenharia de Requisitos (RE): A abordagem sistemática e disciplinada para a especificação e gerenciamento de requisitos com o objetivo de *entender os desejos e necessidades das partes interessadas e minimizar o risco* de entregar um sistema que não atenda a esses desejos e necessidades.

Requisitos
Engenharia

O conceito de *stakeholders* [GIWi2007] é um princípio fundamental da Engenharia de Requisitos (ver Capítulo 2).

Definição 1.3. Parte interessada: Uma pessoa ou organização que influencia os requisitos de um sistema ou que é impactada por esse sistema.

parte interessada

Observe que a influência também pode ser indireta. Por exemplo, algumas partes interessadas podem ter que seguir instruções emitidas por seus gerentes ou organizações.

Seguindo a definição do glossário CPRE RE [Glin2020], usamos o termo *sistema* em sentido amplo neste manual:

Definição 1.4. Sistema: 1. Em geral: um princípio de ordenação e estruturação. 2. Na engenharia: um conjunto coerente e delimitável de elementos que – por ação coordenada – alcançam algum propósito.

Sistema

Observe que um sistema pode compreender outros sistemas ou *componentes* como subsistemas. Os propósitos alcançados por um sistema podem ser entregues por:

- *Implantação* do sistema no(s) local(is) onde é utilizado
- Vender/fornecer o sistema para seus usuários como um *produto*
- Ter provedores que oferecem os recursos do sistema aos usuários como *serviços*

Portanto, usamos o termo sistema como um termo genérico que inclui produtos, serviços, aplicativos ou dispositivos.

1.2 Engenharia de Requisitos: Por que

O desenvolvimento de sistemas (construindo novos, bem como evoluindo os existentes) é um empreendimento caro e constitui um alto risco para todos os participantes. Ao mesmo tempo, os sistemas que têm relevância prática são muito grandes para uma única pessoa compreender intelectualmente. Portanto, os engenheiros desenvolveram vários princípios e práticas para lidar com o risco ao desenvolver um sistema e para dominar o intelectual

complexidade. A Engenharia de Requisitos fornece os princípios e práticas para a perspectiva de requisitos.

A Engenharia de Requisitos (ER) adequada agrega *valor* [Glin2016], [Glin2008] ao processo de desenvolvimento de um sistema:

Valor de RE adequado

- O RE minimiza o risco de falha ou modificações dispendiosas em estágios posteriores de desenvolvimento. A detecção e correção precoce de requisitos errados ou ausentes é muito mais barata do que a correção de erros e retrabalhos causados por requisitos ausentes ou errados em estágios posteriores de desenvolvimento ou mesmo após a implantação de um sistema.
- RE facilita a complexidade intelectual envolvida na compreensão do problema que um sistema deve resolver e na reflexão sobre possíveis soluções.
- RE fornece uma base adequada para estimar esforço e custo de desenvolvimento.
- RE é um pré-requisito para testar o sistema adequadamente.

Os sintomas típicos de ER inadequados são requisitos ausentes, pouco claros ou errados devido a:

Sintomas de RE inadequado

- Equipes de desenvolvimento correndo direto para a implementação de um sistema devido à pressão do cronograma
- Problemas de comunicação entre as partes envolvidas - em particular, entre as partes interessadas e os desenvolvedores do sistema e entre as próprias partes interessadas
- A suposição de que os requisitos são auto-evidentes, o que é errado em a maioria dos casos
- Pessoas que conduzem atividades de ER sem ter educação e habilidades adequadas

1.3 Engenharia de Requisitos: Onde

A Engenharia de Requisitos pode ser aplicada a requisitos para qualquer tipo de sistema.

Aplicação de RE

No entanto, o caso de aplicação dominante para ER hoje envolve sistemas nos quais o software desempenha um papel importante. Esses sistemas consistem em componentes de software, elementos físicos (produtos técnicos, hardware de computação, dispositivos, sensores etc.) e elementos organizacionais (pessoas, cargos, processos de negócios, questões legais e de conformidade etc.).

Os sistemas que contêm software e componentes físicos são chamados de *sistemas cibernetícicos*.

Sistemas ciber-físicos

Sistemas que abrangem software, hardware, pessoas e aspectos organizacionais são chamados de *sistemas sócio-técnicos*.

Sistemas sócio-técnicos

Dependendo da perspectiva adotada, os requisitos ocorrem de várias formas:

Os *requisitos do sistema* descrevem como um sistema deve funcionar e se comportar – conforme observado na interface entre o sistema e seu ambiente – para que o sistema satisfaça os desejos e necessidades de seus stakeholders. No caso de sistemas de software puros, falamos de requisitos de software.

requisitos de sistema

Os *requisitos das partes interessadas* expressam os desejos e necessidades das partes interessadas que devem ser satisfeitos pela construção de um sistema, visto da perspectiva das partes interessadas.

*parte interessada
requisitos*

Os requisitos do usuário são um subconjunto dos requisitos das partes interessadas. Eles cobrem os desejos e necessidades dos usuários de um sistema.

Requisitos do usuário

Os requisitos de domínio especificam as propriedades de domínio necessárias de um sistema sócio-técnico ou ciberfísico.

Requisitos de domínio

Os requisitos de negócios concentram-se nas metas, objetivos e necessidades de negócios de uma organização que devem ser alcançados empregando um sistema (ou uma coleção de sistemas).

Requisitos de negócio

As formas de ocorrência apresentadas acima correspondem às definidas na norma [ISO29148], com exceção dos requisitos de domínio. Pela sua importância, tratamos os requisitos de domínio como uma forma própria. O papel e a importância dos requisitos de domínio são discutidos na Seção 2.2, Princípio 4.

1.4 Engenharia de Requisitos: Como

As principais tarefas em RE são a elicitação (Capítulo 4), documentação (Capítulo 3), validação (Seção 4.4) e gerenciamento (Capítulo 6) de requisitos. O suporte de ferramentas (Capítulo 7) pode ajudar a executar essas tarefas. A análise de requisitos e a resolução de conflitos de requisitos são consideradas parte da elicitação.

Principais tarefas do RE

No entanto, não existe um processo universal que descreva quando e como o ER deve ser executado ao desenvolver um sistema. Para cada desenvolvimento de sistema que precise de atividades de ER, um processo de ER adequado deve ser adaptado a partir de uma ampla gama de possibilidades.

Nenhum processo universal

Fatores que influenciam essa alfaiataria incluem, por exemplo:

- O processo geral de desenvolvimento do sistema - em particular, linear e orientado a planos versus iterativo e ágil O contexto de desenvolvimento - em particular, o
- relacionamento entre o fornecedor, o(s) cliente(s) e os usuários de um sistema A disponibilidade e capacidade das partes interessadas
-

Há também uma dependência mútua entre os produtos de trabalho de requisitos a serem produzidos (consulte a Seção 3.1) e o processo de ER a ser escolhido. Mais detalhes são fornecidos no Capítulo 5.

1.5 O Papel e as Tarefas de um Engenheiro de Requisitos

Na prática, muito poucas pessoas têm o cargo de *Engenheiro de Requisitos*. Consideraremos que as pessoas atuam na *função* de Engenheiro de Requisitos quando:

O Engenheiro de Requisitos é uma função

- Elicitar, documentar, validar e/ou gerenciar requisitos como parte de suas funções
- Ter conhecimento profundo de RE, o que lhes permite definir processos de RE, selecionar práticas de RE apropriadas e aplicar essas práticas adequadamente
- São capazes de preencher a lacuna entre o problema e as possíveis soluções

O papel do Engenheiro de Requisitos faz parte de várias funções de trabalho definidas pelas organizações. Por exemplo, analistas de negócios, especialistas em aplicativos, proprietários de produtos, engenheiros de sistemas e até mesmo desenvolvedores podem atuar na função de Engenheiro de Requisitos. Ter conhecimento e habilidades de ER também é útil para muitos outros profissionais, por exemplo, designers, testadores, arquitetos de sistemas ou CTOs.

1.6 O que aprender sobre Engenharia de Requisitos

O conjunto de habilidades que um Engenheiro de Requisitos deve aprender consiste em vários elementos. Os elementos fundamentais são abordados nos capítulos subsequentes deste manual.

O que você aprenderá neste manual

Além das habilidades técnicas e analíticas, um Engenheiro de Requisitos também precisa do que chamamos de soft skills: a capacidade de ouvir, moderar, negociar e mediar, bem como empatia pelas partes interessadas e abertura às necessidades e ideias dos outros.

RE é regido por um conjunto de princípios fundamentais que se aplicam a todas as tarefas, atividades e práticas de RE. Esses princípios são apresentados no Capítulo 2.

Os requisitos podem ser documentados de várias formas. Vários produtos de trabalho podem ser criados em diferentes níveis de maturidade e detalhe, desde os bastante informais e temporários até produtos de trabalho muito detalhados e estruturados que aderem a regras rígidas de representação. É importante selecionar produtos de trabalho e formas de documentação que sejam adequados para a situação em questão e criar os produtos de trabalho escolhidos adequadamente. Produtos de trabalho e práticas de documentação são apresentados no Capítulo 3.

Os requisitos podem ser elaborados (ou seja, elicitados e validados) com várias práticas. Um Engenheiro de Requisitos deve ser capaz de selecionar as práticas que são mais adequadas em uma determinada situação e aplicá-las adequadamente. As práticas de elaboração são apresentadas no Capítulo 4.

Compreender possíveis processos e estruturas de trabalho permite que os Engenheiros de Requisitos definam uma maneira de trabalhar que se encaixe nas necessidades específicas da situação de desenvolvimento do sistema em questão. Os processos e estruturas de trabalho são apresentados no Capítulo 5.

Os requisitos existentes podem ser gerenciados com várias práticas. Os Engenheiros de Requisitos devem ser capazes de entender quais práticas de gerenciamento de requisitos os suportam para quais tarefas. As práticas de gestão são apresentadas no Capítulo 6.

As ferramentas tornam o ER mais eficiente. Os engenheiros de requisitos precisam saber como as ferramentas de RE podem apoiá-los e como selecionar uma ferramenta adequada para sua situação. O suporte da ferramenta é discutido brevemente no Capítulo 7.

1.7 Leitura Adicional

A terminologia de ER usada neste manual é definida no Glossário CPRE de Terminologia de Engenharia de Requisitos [Glin2020]. Glinz e Wieringa [GIWi2007] explicam a noção de partes interessadas. Lawrence, Wiegers e Ebert [LaWE2001] discutem brevemente os riscos e armadilhas da ER.

Gause e Weinberg [GaWe1989] escreveram um dos primeiros livros sobre ER, que ainda vale a pena dar uma olhada. Pohl [Pohl2010], Robertson e Robertson [RoRo2012] e Wiegers e Beatty [WiBe2013] são livros populares sobre ER. As notas do curso de Glinz [Glin2019] fornecem uma introdução baseada em slides para RE. O livro de van Lamsweerde [vLam2009] apresenta uma abordagem orientada a objetivos para RE. Jackson [Jack1995] contribui com uma coleção perspicaz de ensaios sobre requisitos de software.

Esteja ciente de que o livro oficial do IREB CPRE Foundation Level versão 2.2 [PoRu2015] não está mais totalmente alinhado com a versão 3.0 do CPRE Foundation Level Syllabus, no qual este manual se baseia. No entanto, este livro ainda fornece uma introdução concisa ao RE e será atualizado em breve.

Existem também livros didáticos em outros idiomas além do inglês. Por exemplo, Badreau e Boulanger [BaBo2014] escreveram um livro de ER em francês. Os livros de Ebert [Eber2014] e Rupp [Rupp2014] são livros populares de ER escritos em alemão.

2 Princípios Fundamentais da Engenharia de Requisitos

Neste capítulo, você aprenderá sobre nove princípios básicos da Engenharia de Requisitos (RE).

2.1 Visão Geral dos Princípios

RE é regido por um conjunto de princípios fundamentais que se aplicam a todas as tarefas, atividades e práticas em RE. Uma *tarefa* é um pedaço coerente de trabalho a ser feito (por exemplo, levantamento de requisitos). Uma *atividade* é uma ação ou um conjunto de ações que uma pessoa ou grupo executa para realizar uma tarefa (por exemplo, identificar as partes interessadas ao eliciar requisitos). Uma *prática* é uma forma comprovada de como realizar certos tipos de tarefas ou atividades (por exemplo, usando entrevistas para obter requisitos das partes interessadas).

Os princípios listados na Tabela 2.1 formam a base para as práticas apresentadas nos capítulos subsequentes deste manual.

Tabela 2.1 Nove princípios fundamentais da Engenharia de Requisitos

-
- | | |
|---|---------------------------------------------------------------------------------------------------------------|
| 1 | <i>Orientação de valor:</i> os requisitos são um meio para um fim, não um fim em si |
| 2 | <i>Stakeholders:</i> ER é sobre satisfazer os desejos e necessidades dos stakeholders |
| 3 | <i>Compreensão compartilhada:</i> O desenvolvimento de sistemas bem-sucedidos é impossível sem uma base comum |
| 4 | <i>Contexto:</i> Os sistemas não podem ser entendidos isoladamente |
| 5 | <i>Problema, requisito, solução:</i> um triplô inevitavelmente entrelaçado |
| 6 | <i>Validação:</i> Requisitos não validados são inúteis |
| 7 | <i>Evolução:</i> a mudança de requisitos não é acidental, mas o caso normal |
| 8 | <i>Inovação:</i> Mais do mesmo não é suficiente |
| 9 | <i>Trabalho sistemático e disciplinado:</i> Não podemos prescindir em RE |
-

2.2 Os Princípios Explicados

2.2.1 Princípio 1 – Orientação para o valor: Os requisitos são um meio para um fim, não é um fim em si

O ato de escrever requisitos não é um objetivo em si. Os requisitos são úteis – e o esforço investido na Engenharia de Requisitos é justificado – apenas se agregarem *valor* [Glin2016], [Glin2008], cf. Seção 1.2. Definimos o valor de um requisito como sendo seu *benefício* menos seu *custo*. O benefício de um requisito é o grau em que ele contribui para a construção de sistemas bem-sucedidos (isto é, sistemas que satisfazem os desejos e necessidades de seus stakeholders) e para a redução do risco de falha e retrabalho dispendioso no desenvolvimento do sistema. O custo de um requisito equivale ao custo de extraí-lo, validá-lo, documentá-lo e gerenciá-lo.

Valor dos requisitos: benefício menos custos

Reducir o risco de retrabalho durante o desenvolvimento é parte constituinte do benefício de um requisito bem elaborado. Detectar e corrigir um requisito perdido ou errado durante a implementação ou quando o sistema já está em operação pode facilmente custar uma ou duas ordens de grandeza a mais do que especificar esse requisito corretamente desde o início. Consequentemente, uma quantidade significativa do benefício dos requisitos vem dos custos economizados durante a implementação e operação de um sistema.

Beneficie-se da redução do retrabalho

Em outras palavras, os benefícios de ER geralmente são benefícios de longo prazo, enquanto os custos são imediatos. Isso deve ser levado em consideração na hora de criar um novo projeto. Reduzir custos no curto prazo gastando menos com ER tem um preço: aumenta consideravelmente o risco de retrabalho caro em etapas posteriores do projeto.

O valor da Engenharia de Requisitos pode ser considerado o valor cumulativo dos requisitos especificados. Como os clientes normalmente pagam pelos sistemas a serem implementados, mas não pelos requisitos necessários para fazer isso, o valor econômico de ER é principalmente indireto. Este efeito é reforçado pelo fato de que o benefício dos requisitos decorrentes da redução dos custos de retrabalho é indireto: economiza custos durante a implementação e operação.

*Valor dos Requisitos
Engenharia*

Os efeitos econômicos da Engenharia de Requisitos são principalmente indiretos; RE como tal apenas custos.

Para otimizar o valor de um requisito, os Engenheiros de Requisitos devem encontrar um equilíbrio adequado entre o benefício e o custo de um requisito. Por exemplo, eliciar e documentar a necessidade de uma parte interessada como um requisito facilita a comunicação dessa necessidade entre todas as partes envolvidas. Isso aumenta a probabilidade de que o sistema a ser construído acabe por satisfazer essa necessidade, o que constitui um benefício. Quanto menos ambíguo e mais preciso for o requisito, maior será seu benefício, pois reduz o risco de retrabalho dispendioso devido à má interpretação dos requisitos pelos arquitetos de sistema e equipes de desenvolvimento. Por outro lado, aumentar o grau de não ambigüidade e precisão de um requisito também aumenta o custo envolvido na elicitação e documentação do requisito.

*Otimizando o valor dos
requisitos*

Na verdade, a quantidade de RE necessária para atingir os requisitos com valor ótimo depende de vários fatores dados pela situação específica em que os requisitos estão sendo criados e usados. Obviamente, o risco de construir um sistema que eventualmente não satisfaça os desejos e necessidades de seus stakeholders, o que pode resultar em falha ou retrabalho dispendioso, é a *força motriz* que determina a quantidade de RE necessária. Em primeiro lugar, a criticidade de cada requisito deve ser avaliada em termos da importância da(s) parte(s) interessada(s) que declara(m) o requisito (consulte o Princípio 2) e o impacto de não cumprir o requisito (Figura 2.1).

Fatores de influência

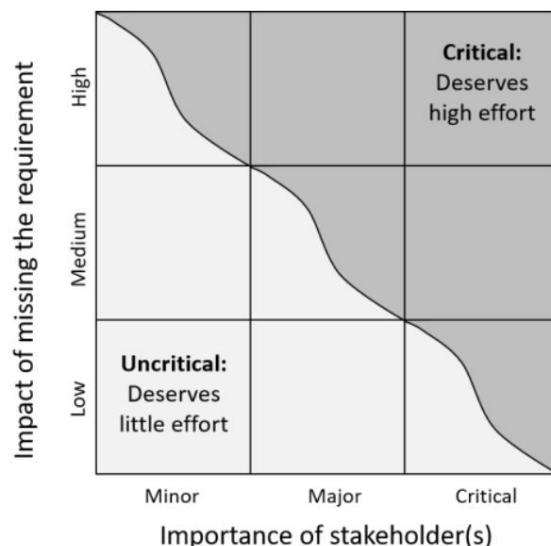


Figura 2.1 Avaliando a criticidade de um requisito [Glin2008]

Além disso, os seguintes fatores de influência devem ser considerados:

- Esforço necessário para especificar o requisito
- Distinção do requisito (o quanto ele contribui para o sucesso do sistema geral)

- Grau de entendimento compartilhado entre as partes interessadas e desenvolvedores e entre as partes interessadas
- Existência de sistemas de referência (que possam servir de especificação a título de exemplo)
- Duração do ciclo de feedback (o tempo entre errar um requisito e detectar o erro)

- Tipo de relação cliente-fornecedor
- Conformidade regulamentar necessária

Resumimos esta questão em duas regras práticas:

- A quantidade ideal de RE a ser investida depende da situação específica e é determinada por muitos fatores influentes.
- O esforço investido em RE deve ser inversamente proporcional ao risco que você está disposto a correr.

2.2.2 Princípio 2 – Stakeholders: ER é sobre a satisfação dos desejos e necessidades dos stakeholders

O objetivo final da construção de um sistema é que o sistema, quando usado, resolva os problemas que seus usuários precisam resolver e satisfaça as expectativas de outras pessoas - por exemplo, aqueles que solicitaram e pagaram pelo sistema, ou aqueles que estão responsável pela segurança na organização que utiliza o sistema. Portanto, temos que descobrir as necessidades e expectativas das pessoas que têm interesse no sistema, os *stakeholders do sistema* [GIWI2007]. Os principais objetivos do RE são *entender os desejos e necessidades das partes interessadas e minimizar o risco* de entregar um sistema que não atenda a esses desejos e necessidades; ver Definição 1.2 na Seção 1.2.

Cada parte interessada tem um *papel* no contexto do sistema a ser construído – por exemplo, usuário, cliente, operador ou regulador. Dependendo do processo de ER usado, os desenvolvedores de um sistema também podem ser partes interessadas. Este é frequentemente o caso no desenvolvimento ágil e orientado para o mercado. Uma parte interessada também pode ter mais de uma função ao mesmo tempo. Para cada função de parte interessada relevante, pessoas adequadas que atuam nessa função devem ser selecionadas como representantes.

papéis das partes interessadas

Para papéis de partes interessadas com muitos indivíduos ou quando os indivíduos são desconhecidos, *personas* (personagens fictícios que representam um grupo de usuários com características semelhantes) podem ser definidas como um substituto. Para sistemas que já estão em uso, os usuários que fornecem feedback sobre o sistema ou solicitam novos recursos também devem ser considerados partes interessadas.

Pessoas

Faz sentido classificar as partes interessadas em três categorias com relação ao grau de influência que uma parte interessada tem sobre o sucesso do sistema:

Classificando as partes interessadas

- *Crítico*: não considerar essas partes interessadas resultará em problemas graves e provavelmente fará com que o sistema falhe ou o inutilize.
- *Maior*: não considerar essas partes interessadas terá um impacto adverso no sucesso do sistema, mas não o fará falhar.

- Menor: não considerar essas partes interessadas terá pouca ou nenhuma influência no sucesso do sistema.

Essa classificação é útil ao avaliar a criticidade de um requisito (consulte a Figura 2.1) e ao negociar conflitos entre as partes interessadas (consulte abaixo).

Não é suficiente considerar apenas os requisitos dos usuários finais e clientes.

Fazer isso significaria que poderíamos perder requisitos críticos de outras partes interessadas, o que pode facilmente levar a projetos de desenvolvimento que falham ou excedem seus orçamentos e prazos.

Considerar apenas usuários finais e clientes não é suficiente

Envolver as pessoas certas nas funções relevantes das partes interessadas é crucial para o sucesso da ER.

As práticas para identificar, priorizar e trabalhar com as partes interessadas são discutidas no Capítulo 4.

As partes interessadas em diferentes papéis naturalmente têm diferentes pontos de vista [NuKF2003] de um sistema a ser desenvolvido. Por exemplo, os usuários geralmente desejam um sistema para dar suporte às suas tarefas de maneira otimizada, os gerentes que solicitam o sistema desejam obtê-lo a um custo razoável e o diretor de segurança da organização se preocupa principalmente com a segurança do sistema. Mesmo as partes interessadas na mesma função podem ter necessidades diferentes.

As partes interessadas têm pontos de vista diferentes

Por exemplo, no grupo de usuários finais, os usuários casuais têm requisitos de interface de usuário que podem diferir bastante dos usuários profissionais.

Como consequência, não é suficiente apenas coletar requisitos das partes interessadas.

Gerenciamento de inconsistências e conflitos

É vital identificar inconsistências e conflitos entre os requisitos de diferentes stakeholders e resolvê-los, seja por consenso, overruling ou especificando variantes do sistema para stakeholders que de fato têm necessidades diferentes; consulte a Seção 4.3.

2.2.3 Princípio 3 – Compreensão compartilhada: O desenvolvimento bem-sucedido de sistemas é impossível sem uma base comum

O desenvolvimento do sistema, incluindo RE, é um esforço de várias pessoas. Para tornar tal empreendimento um sucesso, as pessoas envolvidas precisam de um *entendimento compartilhado* do problema e dos requisitos que dele decorrem [GIFr2015].

A necessidade de compreensão compartilhada

O RE cria, promove e assegura o entendimento compartilhado entre as partes envolvidas: partes interessadas, engenheiros de requisitos e desenvolvedores. Distinguimos entre duas formas de entendimento compartilhado:

- O entendimento *compartilhado explícito* é alcançado por meio de requisitos cuidadosamente eliciados, documentados e acordados. Este é o principal objetivo do RE em processos orientados a planos.
- O entendimento *compartilhado implícito* é baseado no conhecimento compartilhado sobre necessidades, visões, contexto, etc. Em ER ágil, quando os requisitos não são totalmente especificados por escrito, a confiança no entendimento compartilhado implícito é fundamental.

Entendimento compartilhado explícito e implícito

Tanto o entendimento compartilhado implícito quanto o explícito podem ser *falsos*, o que significa que as pessoas acreditam que têm um entendimento compartilhado de um problema, mas na verdade interpretam esse problema de maneiras diferentes. Portanto, nunca podemos confiar cegamente no entendimento compartilhado. Em vez disso, a tarefa de ER é criar e fomentar o entendimento compartilhado e também protegê-lo – ou seja, avaliar se existe um verdadeiro entendimento compartilhado. Para limitar o esforço

envolvidos, é vital concentrar-se no entendimento compartilhado sobre coisas *relevantes* - ou seja, aqueles aspectos que estão dentro do limite de contexto de um sistema (cf. Princípio 4).

Mesmo com um entendimento compartilhado perfeito, requisitos importantes ainda podem ser perdidos porque ninguém os considerou. A Figura 2.2 ilustra diferentes situações de entendimento compartilhado com um exemplo simples de um casal que deseja instalar um balanço em seu jardim para seus filhos [Glin2019]. A nota adesiva no meio simboliza uma especificação escrita.

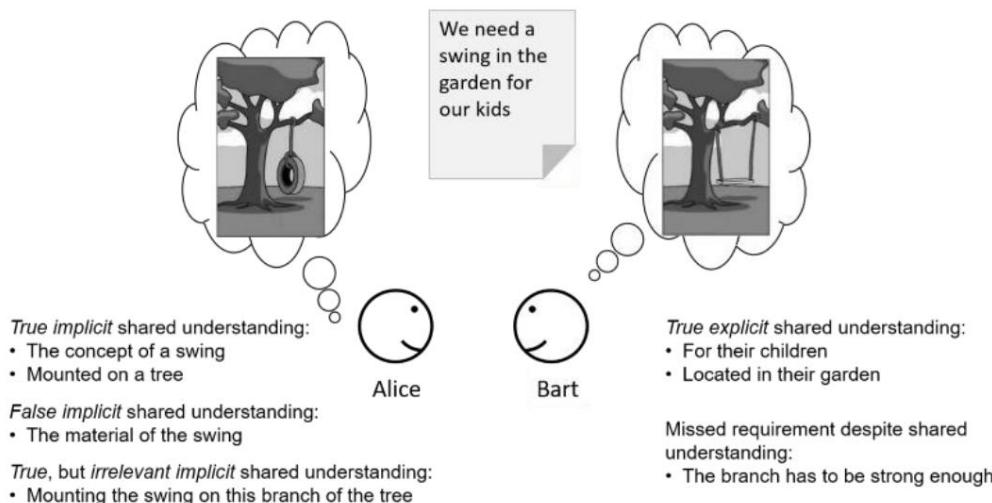


Figura 2.2 Diferentes situações de compreensão compartilhada – ilustradas com o exemplo de um casal que deseja instalar um balanço para seus filhos

Práticas comprovadas para *alcançar* o entendimento compartilhado incluem trabalhar com glossários (Seção 3.5), criar protótipos (Seção 3.7) ou usar um sistema existente como ponto de referência.

Alcançar o entendimento compartilhado em ER

O principal meio para *avaliar* o verdadeiro entendimento compartilhado explícito em ER é validar completamente todos os requisitos especificados (cf. Princípio 6 e Seção 4.4). As práticas para avaliar o entendimento compartilhado implícito incluem fornecer exemplos de resultados esperados, construir protótipos ou estimar o custo de implementação de um requisito.

Avaliando o entendimento compartilhado em ER

A prática mais importante para reduzir o impacto do falso entendimento compartilhado é usar um processo com ciclos de feedback curtos (Capítulo 5).

Existem fatores que constituem facilitadores ou obstáculos do entendimento compartilhado. Por exemplo, os facilitadores são:

Facilitadores e obstáculos

- ▶ Conhecimento de domínio
- ▶ Padrões específicos de domínio
- ▶ Colaboração bem-sucedida anterior
- ▶ Existência de sistemas de referência conhecidos por todas as pessoas envolvidas
- ▶ Cultura e valores compartilhados
- ▶ Confiança mútua informada (não cega!)

Obstáculos são:

- ▶ distância geográfica
- ▶ Relação fornecedor-cliente pautada pela desconfiança mútua
- ▶ Terceirização
- ▶ Restrições regulatórias

- Equipes grandes e diversificadas
- Alta rotatividade entre as pessoas envolvidas

Quanto menor a probabilidade e o impacto do falso entendimento compartilhado e melhor a proporção entre facilitadores e obstáculos, mais RE pode confiar no entendimento compartilhado implícito. Por outro lado, quanto menos facilitadores e mais obstáculos para o entendimento compartilhado tivermos e quanto maior o risco e o impacto do falso entendimento compartilhado para um requisito, mais tais requisitos terão que ser especificados e validados explicitamente.

Confando no entendimento compartilhado

2.2.4 Princípio 4 – Contexto: Sistemas não podem ser entendidos isoladamente

Requisitos nunca vêm de forma isolada. Eles se referem a *sistemas* que estão inseridos em um *contexto*. Enquanto o termo *contexto* em geral denota a rede de pensamentos e significados necessários para a compreensão de fenômenos ou enunciados, ele tem um significado especial em RE.

Definição 2.1. Contexto (em RE): A parte do ambiente de um sistema que é relevante para a compreensão do sistema e seus requisitos.

Contexto

O contexto de um sistema é delimitado pelo limite do sistema e pelo limite do contexto [Pohl2010] (ver Figura 2.3).

Definição 2.2. Limite de contexto: O limite entre o contexto de um sistema e as partes do domínio do aplicativo que são irrelevantes para o sistema e seus requisitos.

limite de contexto

A *fronteira de contexto* separa a parte relevante do ambiente de um sistema a ser desenvolvida da parte irrelevante, ou seja, a parte que não influencia o sistema a ser desenvolvido e, portanto, não precisa ser considerada durante a Engenharia de Requisitos.

Definição 2.3. Limite do sistema: O limite entre um sistema e seu contexto circundante.

Limite do sistema

O *limite do sistema* delimita o sistema como ele será após sua implementação e implantação. O limite do sistema geralmente não é claro inicialmente e pode mudar com o tempo. Esclarecer os limites do sistema e definir as interfaces externas entre um sistema e os elementos em seu contexto são tarefas genuínas de ER.

O limite do sistema frequentemente coincide com o *escopo* do desenvolvimento de um sistema.

Definição 2.4. Escopo: A gama de coisas que podem ser moldadas e projetadas ao desenvolver um sistema.

Escopo

Às vezes, no entanto, o limite do sistema e seu escopo não correspondem (consulte a Figura 2.3). Pode haver componentes dentro do limite do sistema que precisam ser reutilizados como estão (ou seja, não podem ser moldados ou projetados), o que significa que estão fora do escopo. Por outro lado, pode haver coisas no contexto do sistema que podem ser reprojetadas quando o sistema é desenvolvido, o que significa que estão dentro do escopo.

Como as interfaces externas residem no limite do sistema, RE deve determinar quais dessas interfaces estão no escopo (ou seja, elas podem ser moldadas e projetadas no processo de desenvolvimento) e quais são dadas e fora do escopo.

Não é suficiente considerar apenas os requisitos dentro do limite do sistema.

Primeiro, quando o escopo inclui partes do contexto do sistema, conforme mostrado na Figura 2.3, as mudanças de contexto dentro do escopo podem afetar os requisitos do sistema. Por exemplo, quando um processo de negócio deve ser parcialmente automatizado por um sistema, pode ser útil adaptar o processo para simplificar sua automação. Obviamente, tal adaptação impacta os requisitos do sistema.

Considerando mudanças de contexto em RE

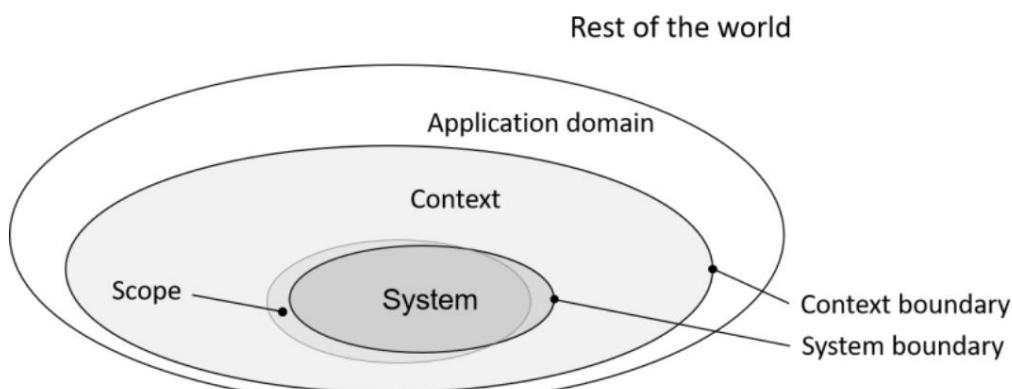


Figura 2.3 Sistema, contexto e escopo

Em segundo lugar, pode haver fenômenos do mundo real no contexto do sistema que um sistema deve monitorar ou controlar. Os requisitos para tais fenômenos devem ser declarados como requisitos de domínio e devem ser adequadamente mapeados para os requisitos do sistema. Por exemplo, em um carro equipado com caixa de câmbio automática, é necessário que a posição de estacionamento possa ser engatada somente quando o carro não estiver em movimento. No contexto de um sistema de software que controla a caixa de câmbio, esse é um requisito de domínio. Para atender a esse requisito, o controlador precisa saber se o carro está ou não em movimento. No entanto, o controlador não pode detectar esse fenômeno diretamente. Portanto, o fenômeno do mundo real “o carro não está se movendo” deve ser mapeado para um fenômeno que o sistema de controle possa detectar – por exemplo, a entrada de um sensor que cria pulsos quando uma roda do carro está girando. O requisito de domínio relativo ao acionamento da posição de estacionamento é então mapeado para um requisito do sistema, como “O sistema de controle da caixa de câmbio deve permitir o acionamento da posição de estacionamento somente se nenhum pulso for recebido dos sensores de giro das rodas”.

Mapeando fenômenos do mundo real

Em terceiro lugar, pode haver requisitos que não podem ser satisfeitos por nenhuma implementação de sistema, a menos que certos *requisitos de domínio* e *suposições de domínio* no contexto do sistema sejam válidos. Suposições de domínio são suposições sobre fenômenos do mundo real no contexto de um sistema. Por exemplo, considere um sistema de controle de tráfego aéreo (ATS). O requisito “R1: O ATS deve manter posições precisas para todas as aeronaves controladas pelo sistema” é um requisito importante do sistema. No entanto, este requisito pode ser atendido apenas se o radar no contexto do ATS satisfizer os requisitos de identificar corretamente todas as aeronaves no espaço aéreo controlado pelo radar e determinar corretamente sua posição. Por sua vez, esses requisitos podem ser satisfeitos apenas se todas as aeronaves detectadas pelo radar responderem adequadamente aos sinais de interrogação enviados pelo radar.

Requisitos de domínio e domínio premissas

Além disso, o requisito R1 pode ser atendido apenas se certas suposições de domínio no contexto do ATS forem mantidas - por exemplo, que o radar não seja bloqueado por um invasor mal-intencionado e que nenhuma aeronave esteja voando a uma altitude inferior à que o radar pode detectar .

RE vai além de considerar os requisitos dentro do limite do sistema e definir as interfaces externas no limite do sistema. RE também deve lidar com fenômenos no contexto do sistema.

Consequentemente, ER também deve considerar questões no contexto do sistema:

- Se mudanças no contexto podem ocorrer, como elas impactam os requisitos do sistema?
- Quais requisitos no contexto do mundo real são relevantes para o sistema a ser desenvolvido?
- Como esses requisitos do mundo real podem ser mapeados adequadamente para os requisitos do sistema?
- Quais suposições sobre o contexto devem ser mantidas para que o sistema funcione adequadamente e os requisitos do mundo real sejam atendidos?

RE tem que considerar o contexto

2.2.5 Princípio 5 – Problema, requisito, solução: Uma relação inevitavelmente triplo

Problemas, suas soluções e requisitos estão intimamente e inevitavelmente interligados [SwBa1982]. Toda situação em que as pessoas não estão satisfeitas com a maneira como estão fazendo as coisas pode ser considerada como a ocorrência de um *problema*. Para eliminar esse problema, um sistema sócio-técnico pode ser desenvolvido e implantado. Os *requisitos* para esse sistema devem ser capturados para tornar o sistema uma *solução* eficaz para o problema. Especificar requisitos não faz sentido se não houver nenhum problema a ser resolvido ou se nenhuma solução for desenvolvida. Também não faz sentido desenvolver uma solução que esteja em busca de um problema para resolver ou de requisitos a satisfazer.

Por que problemas, requisitos e soluções são entrelaçados

É importante observar que problemas, requisitos e soluções não ocorrem necessariamente nessa ordem. Por exemplo, ao projetar um sistema inovador, as ideias de solução criam necessidades do usuário que devem ser trabalhadas como requisitos e implementadas em uma solução real.

Problemas, requisitos e soluções podem ser interligados de várias maneiras:

Formas de entrelaçamento

- Entrelaçamento hierárquico: ao desenvolver grandes sistemas com uma hierarquia multinível de subsistemas e componentes, requisitos de alto nível levam a decisões de projeto de alto nível, que por sua vez informam requisitos de nível inferior que levam a decisões de projeto de nível inferior, etc.
- Viabilidade técnica: especificar requisitos não factíveis é um desperdício de esforço; no entanto, pode ser possível avaliar a viabilidade de um requisito apenas ao explorar soluções técnicas.
- Validação: os protótipos, que são um poderoso meio de validação de requisitos, constituem soluções parciais do problema.
- Viés de solução: diferentes partes interessadas podem vislumbrar soluções diferentes para um determinado problema, com a consequência de especificar requisitos diferentes e conflitantes para esse problema.

O entrelaçamento de problemas, requisitos e soluções também traz consequências para o processo de desenvolvimento de um sistema:

Consequências do entrelaçamento

- Raramente é possível separar estritamente ER das atividades de projeto e implementação do sistema. Portanto, processos estritos de desenvolvimento em cascata não funcionam bem.
- No entanto, os Engenheiros de Requisitos visam separar problemas, requisitos e soluções uns dos outros tanto quanto possível ao pensar, comunicar e documentar. Essa *separação de preocupações* torna as tarefas de ER mais fáceis de lidar.

Apesar do inevitável entrelaçamento de problemas, requisitos e soluções, os Engenheiros de Requisitos se esforçam para separar as preocupações com os requisitos das preocupações com a solução ao pensar, comunicar e documentar.

2.2.6 Princípio 6 – Validação: Requisitos não validados são inúteis

Quando um sistema é desenvolvido, o sistema final implantado deve satisfazer os desejos e necessidades das partes interessadas. No entanto, realizar essa verificação no final do desenvolvimento é muito arriscado. A fim de controlar o risco de partes interessadas insatisfeitas desde o início, a validação dos requisitos deve começar durante a ER (ver Figura 2.4).

Por que precisamos de validação antecipada

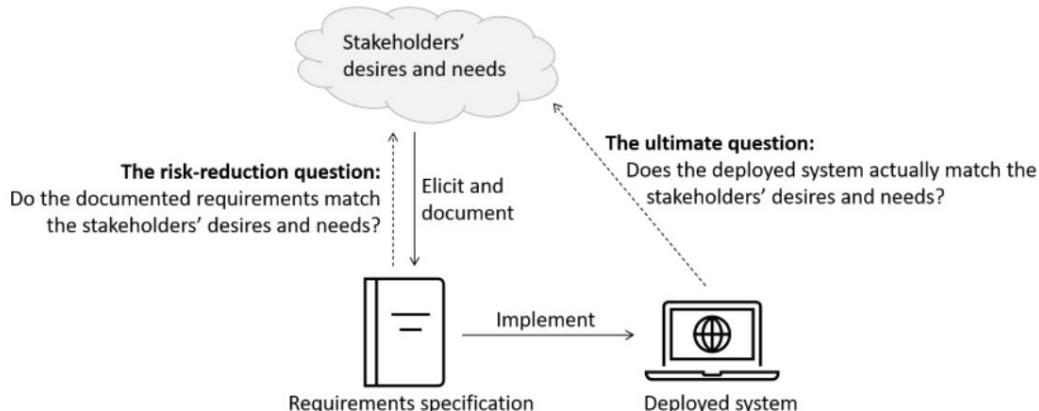


Figura 2.4 Validação [Glin2019]

Definição 2.5. Validação: O processo de confirmação de que um item (um sistema, um produto de trabalho ou parte dele) atende às necessidades de seus stakeholders.

Validação

Em RE, a *validação* é o processo de confirmação de que os requisitos documentados atendem às necessidades das partes interessadas; em outras palavras, confirmando se os requisitos corretos foram especificados.

A validação é uma atividade central em ER: não há especificação sem validação.

Ao validar os requisitos, temos que verificar se:

Coisas para validar

- O acordo sobre os requisitos foi alcançado entre as partes interessadas (conflitos resolvidos, prioridades definidas)

- Os desejos e necessidades das partes interessadas são adequadamente cobertos pelos requisitos
- As suposições de domínio (consulte o Princípio 4 acima) são razoáveis - ou seja, podemos esperar que essas suposições possam ser atendidas quando o sistema for implantado e operado

Práticas para validação de requisitos são discutidas na Seção 4.4.

2.2.7 Princípio 7 – Evolução: A mudança de requisitos não é acidental, mas o caso normal

Todo sistema técnico está sujeito à evolução. Necessidades, negócios e capacidades mudam continuamente. Como consequência natural, os requisitos para sistemas que devem satisfazer necessidades, dar suporte a negócios e usar recursos técnicos também mudarão. Caso contrário, tais sistemas e seus requisitos perdem progressivamente seu valor e eventualmente se tornam inúteis.

A evolução é inevitável

Um requisito pode mudar enquanto os Engenheiros de Requisitos ainda estão levantando outros requisitos, quando o sistema está em implementação ou quando está implantado e sendo usado.

São muitos os motivos que levam a solicitações de alteração de um requisito ou conjunto de requisitos de um sistema, por exemplo:

Razões para alterar os requisitos

- Processos de negócios alterados
- Concorrentes lançando novos produtos ou serviços
- Clientes mudando suas prioridades ou opiniões
- Mudanças na tecnologia
- Feedback dos usuários do sistema solicitando recursos novos ou alterados
- Detecção de erros em requisitos ou detecção de suposições de domínio defeituosas

Os requisitos também podem mudar devido ao feedback das partes interessadas ao validar requisitos, devido à detecção de falhas em requisitos previamente eliciados ou devido a necessidades alteradas.

Como consequência, os Engenheiros de Requisitos devem perseguir dois objetivos aparentemente contraditórios:

Habilitando a mudança enquanto preserva a estabilidade

- Permitir que os requisitos mudem, porque tentar ignorar a evolução dos requisitos seria inútil.
- Mantenha os requisitos estáveis, porque sem alguma estabilidade nos requisitos, o custo da mudança pode se tornar proibitivamente alto. Além disso, as equipes de desenvolvimento não podem se desenvolver sistematicamente se os requisitos mudarem diariamente.

Os engenheiros de requisitos precisam gerenciar a evolução dos requisitos. Caso contrário, a evolução os administrará.

Os processos de mudança para requisitos que tratam de ambos os objetivos são discutidos na Seção 6.7.

2.2.8 Princípio 8 – Inovação: Não basta mais do mesmo

Enquanto o RE está preocupado em satisfazer os desejos e necessidades dos stakeholders, os Engenheiros de Requisitos que apenas desempenham o papel de gravador de voz dos stakeholders, especificando exatamente o que os stakeholders dizem, estão fazendo o trabalho errado. Dar às partes interessadas exatamente o que elas querem significa perder a oportunidade de fazer as coisas melhor do que antes.

*Engenheiros de Requisitos
não são gravadores de voz*

Por exemplo, imagine o seguinte cenário. Uma seguradora deseja renovar o sistema de relatórios para seus agentes. O relatório mais utilizado é uma tabela com 18 colunas, que tem cerca de duas vezes a largura da tela quando exibida nos laptops dos agentes. A visualização desse relatório requer muita rolagem. As partes interessadas, portanto, desejam ampliar o relatório, usando os botões de mais e menos na tela. Nesta situação, bons Engenheiros de Requisitos não irão apenas registrar isso como um requisito. Em vez disso, eles começarão a fazer perguntas. Acontece que a empresa vai substituir os laptops dos agentes por tablets. Portanto, implementar gestos com dois dedos em vez dos botões necessários tornará o zoom muito mais fácil. Além disso, verifica-se que três colunas do relatório podem ser eliminadas com uma pequena alteração nas regras de relatórios, que a empresa concorda em fazer. Além disso, apenas seis colunas do relatório são sempre necessárias; as colunas restantes são usadas apenas em casos especiais.

*Exemplo de como poderia
funcionar*

Levando isso em consideração, os Engenheiros de Requisitos sugerem que as partes interessadas exijam que (1) o relatório mostre as mesmas informações do sistema atual, menos o conteúdo das três colunas eliminadas; (2) quando o relatório é aberto, apenas as seis colunas importantes são exibidas na largura total, enquanto as outras colunas são reduzidas à largura mínima; e (3) que os agentes podem expandir uma coluna recolhida tocando em seu cabeçalho (e recolhê-la novamente com outro toque).

Dessa forma, os agentes obterão um sistema que não apenas adiciona uma solução alternativa para visualizar um relatório superdimensionado. Em vez disso, o sistema resolverá o problema dos agentes com um recurso inovador para filtrar informações e também contará com um meio intuitivo de zoom.

É assim que surge a inovação. Bons engenheiros de requisitos estão atentos à inovação: eles se esforçam não apenas para satisfazer as partes interessadas, mas também para deixá-las felizes, entusiasmadas ou seguras [KSTT1984]. Ao mesmo tempo, evitam a armadilha de acreditar que sabem tudo melhor do que os stakeholders.

Conscientização da inovação

Bons Engenheiros de Requisitos vão além do que seus stakeholders lhes dizem.

Em pequena escala, a RE molda sistemas inovadores, buscando novos recursos empolgantes e facilidade de uso. Além disso, os Engenheiros de Requisitos também precisam olhar para o quadro geral, explorando com as partes interessadas se existem maneiras disruptivas de fazer as coisas, levando à inovação em larga escala [MaGR2004].

*Moldando sistemas
inovadores em ER*

A Seção 0 discute várias técnicas para fomentar a inovação em ER.

2.2.9 Princípio 9 – Trabalho sistemático e disciplinado: Não podemos prescindir em RÉ

O ER não é uma arte, mas uma disciplina, que exige que o ER seja executado de forma sistemática e disciplinada. Independentemente do(s) processo(s) usado(s) para desenvolver um sistema, precisamos empregar processos e práticas de ER adequados para eliciar sistematicamente,

*Como disciplina, ER exige
um trabalho sistemático e
disciplinado*

documentar, validar e gerenciar requisitos. Mesmo quando um sistema é desenvolvido de maneira ad hoc, uma abordagem sistemática e disciplinada de ER (por exemplo, promovendo sistematicamente o entendimento compartilhado, consulte o Princípio 3) melhorará a qualidade do sistema resultante.

Agilidade e flexibilidade não são desculpas válidas para um estilo de trabalho não sistemático e ad hoc em ER.

No entanto, não existe um processo de ER universal nem um conjunto universal de práticas de ER que funcionem bem em todas as situações ou pelo menos na maioria das situações: não existe um “tamanho único” em ER.

O trabalho sistemático e disciplinado significa que os Engenheiros de Requisitos:

- Configure um processo de RE que seja adequado para o problema em questão e que se encaixe bem no processo usado para desenvolver o sistema (consulte o Capítulo 5).
- Do conjunto de práticas de ER e produtos de trabalho disponíveis, selecione aqueles que são mais adequados para determinado problema, contexto e ambiente de trabalho (consulte os Capítulos 3, 4 e 6).
- Nem sempre use o mesmo processo, práticas e produtos de trabalho.
- Não reutilize processos e práticas de trabalhos anteriores de RE bem-sucedidos sem reflexão.

Não há “tamanho único”

Quais Requisitos

Engenheiros precisam fazer

2.3 Leitura Adicional

Glinz [Glin2008] discute o valor dos requisitos de qualidade e dos requisitos em geral [Glin2016].

Glinz e Wieringa [GIWi2007] explicam a noção e a importância dos stakeholders.

Glinz e Fricker [GIFr2015] discutem o papel e a importância do entendimento compartilhado.

Os artigos de Jackson [Jack1995b] e Gunter et al. [GGJZ2000] são fundamentais para o problema de requisitos em contexto. O papel do contexto em RE também é discutido por Pohl [Pohl2010].

Gause e Weinberg [GaWe1989] discutem a interdependência de problemas e soluções. Swartout e Balzer [SwBa1982] foram os primeiros a apontar que raramente é possível criar uma especificação completa antes de iniciar a implementação.

A validação é abordada em qualquer livro de RE. Grünbacher e Seyff [GrSe2005] discutem como chegar a um acordo negociando os requisitos.

Kano et al. [KSTT1984] foram os primeiros a enfatizar o papel da inovação. Maalej, Nayebi, Johann e Ruhe [MNJR2016] discutem o uso de feedback explícito e implícito do usuário para RE. Maiden, Gitzikis e Robertson [MaGR2004] discutem como a criatividade pode fomentar a inovação em ER. Gorscheck et al. [GFpk2010] descrevem um processo sistemático de inovação.

3 Produtos de Trabalho e Práticas de Documentação

A Engenharia de Requisitos (RE) tradicional exige a redação de uma especificação de requisitos abrangente, completa e inequívoca [IEEE830], [Glin2016]. Embora ainda seja apropriado criar especificações de requisitos completas em muitos casos, também há muitos outros casos em que o custo de escrever tais especificações excede seu benefício. Por exemplo, especificações de requisitos completos são úteis ou até mesmo necessárias ao licitar ou terceirizar o projeto e a implementação de um sistema ou quando um sistema é crítico para a segurança e a conformidade regulamentar é necessária. Por outro lado, onde as partes interessadas e os desenvolvedores unem forças para definir e desenvolver um sistema iterativamente, escrever uma especificação de requisitos abrangente não faz sentido. Portanto, é vital em ER adaptar a documentação ao contexto do projeto e selecionar produtos de trabalho para documentar requisitos e informações relacionadas a requisitos que gerem valor ideal para o projeto.

Neste capítulo, você aprenderá sobre os produtos de trabalho RE típicos e como criá-los.

3.1 Produtos de Trabalho na Engenharia de Requisitos

Há uma variedade de produtos de trabalho que são usados em RE.

Definição 3.1. Produto de trabalho: Um resultado intermediário ou final registrado gerado em um processo de trabalho.

produto de trabalho

Consideramos o termo **artefato** como sinônimo de produto de trabalho. Preferimos o termo produto de trabalho a artefato para expressar a conotação de que um produto de trabalho é o resultado do trabalho realizado em um processo de trabalho.

Artefato

De acordo com essa definição, um produto de trabalho de RE pode ser qualquer coisa que expresse requisitos, desde uma única frase ou diagrama até uma especificação de requisitos de sistema que abrange centenas de páginas. Também é importante observar que um produto de trabalho pode conter outros produtos de trabalho.

3.1.1 Características dos Produtos de Trabalho

Os produtos de trabalho podem ser caracterizados pelas seguintes facetas: *finalidade*, *tamanho*, *representação*, *vida útil* e *armazenamento*.

Caracterização de produtos de trabalho

A Tabela 3.1 fornece uma visão geral dos produtos de trabalho típicos usados em RE junto com sua respectiva finalidade (ou seja, o que o produto de trabalho especifica ou fornece) e tamanho típico. A tabela está estruturada em quatro grupos: produtos de trabalho para requisitos únicos, conjuntos coerentes de requisitos, documentos ou estruturas de documentação e outros produtos de trabalho.

Existem muitas maneiras diferentes de representar um produto de trabalho. Em RE, as representações baseadas em *linguagem natural*, *modelos* e *modelos* são de particular importância. Estes são discutidos nas Seções 3.2, 3.3 e 3.4, respectivamente. Existem outras representações, como desenhos ou protótipos, que são abordadas na Seção 3.7.

Representação

Todo produto de trabalho tem uma *vida útil*. Este é o período de tempo desde a criação do produto de trabalho até o ponto em que o produto de trabalho é descartado ou se torna irrelevante. Distinguimos entre três categorias de produtos de trabalho com relação ao tempo de vida: produtos de trabalho *temporários*, *evolutivos* e *duráveis*.

Vida útil

Produtos de trabalho temporários são criados para dar suporte à comunicação e criar entendimento compartilhado (por exemplo, um esboço de uma interação usuário-sistema criada em uma oficina). Os produtos de trabalho temporário são descartados após o uso; nenhum metadado é mantido sobre esses produtos de trabalho.

produtos de trabalho temporário

Os produtos de trabalho em evolução surgem em várias iterações ao longo do tempo (por exemplo, uma coleção de histórias de usuários que cresce tanto no número de histórias quanto no conteúdo da história). Alguns metadados (pelo menos o proprietário, status e histórico de revisão) devem ser mantidos para cada produto de trabalho em evolução. Dependendo da importância e do status de um produto de trabalho, os procedimentos de controle de mudança precisam ser aplicados ao modificar um produto de trabalho em evolução.

Produtos de trabalho em evolução

Produtos de trabalho duráveis foram definidos ou lançados (por exemplo, uma especificação de requisitos que faz parte de um contrato ou um sprint backlog que é implementado em uma determinada iteração). Um conjunto completo de metadados deve ser mantido para gerenciar o produto de trabalho adequadamente e um processo de alteração elaborado deve ser seguido para alterar um produto de trabalho durável (Capítulo 6).

produtos de trabalho duráveis

Um produto de trabalho temporário pode se tornar um produto em evolução quando os Engenheiros de Requisitos decidem manter um produto de trabalho e desenvolvê-lo ainda mais. Neste caso, alguns metadados devem ser adicionados para manter a evolução do produto de trabalho sob controle. Quando um produto de trabalho em evolução é baseado ou lançado, ele muda seu status de vida útil de evolutivo para durável.

Temporário | Evoluindo | Durável

Tabela 3.1 Visão geral dos produtos de trabalho RE

produto de trabalho	Finalidade: O produto de trabalho especifica/fornecida	Tamanho*	Produtos de trabalho RE típicos
Requisitos únicos			
Requisito individual	Um único requisito, geralmente em forma de texto	S	
História do usuário	Uma função ou comportamento do ponto de vista de uma parte interessada	S	
Conjuntos coerentes de requisitos			
Caso de uso	Uma função do sistema da perspectiva de um ator ou usuário	SM	
modelo gráfico	Vários aspectos, por exemplo, contexto, função, comportamento (consulte a Seção 3.4)	M	
Descrição da tarefa	Uma tarefa que um sistema deve realizar	SM	
Descrição da interface externa	As informações trocadas entre um sistema e um ator no contexto do sistema	M	
Épico	Uma visão de alto nível da necessidade de uma parte interessada	M	
Recurso	Uma característica distintiva de um sistema	SM	
Documentos ou estruturas de documentação			
Especificação dos requisitos do sistema**	Um documento de requisitos abrangente	L-XL	
Backlog de produto e sprint	Uma lista de itens de trabalho, incluindo requisitos	ML	
mapa da história	Um arranjo visual de histórias de usuários	M	
Visão	Uma imaginação conceitual de um sistema futuro	M	
Outros produtos de trabalho			
Glossário	Terminologia comum inequívoca e acordada	M	
Nota textual ou esboço gráfico	Um memorando para comunicação e compreensão	S	
Protótipo	Uma especificação por exemplo, especialmente para entender, validar e negociar sobre requisitos	SL	

*: S: Pequeno, M: Médio, L: Grande, XL: Muito grande

**: Outros exemplos são: especificação de requisitos de negócios, especificação de requisitos de domínio, especificação de requisitos de partes interessadas/usuários ou especificação de requisitos de software

Atualmente, a maioria dos produtos de trabalho são armazenados eletronicamente como arquivos, em bancos de dados ou em ferramentas de RE. Produtos de trabalho informais e temporários também podem ser armazenados em outras mídias – por exemplo, papel ou notas adesivas em um quadro Kanban.

Armazenando produtos de trabalho

3.1.2 Níveis de Abstração

Os requisitos e seus produtos de trabalho correspondentes ocorrem em vários níveis de abstração - desde, por exemplo, requisitos de alto nível para um novo processo de negócios até requisitos em um nível muito detalhado, como a reação de um componente de software específico a um evento excepcional.

Os requisitos ocorrem em vários níveis de abstração

Requisitos de negócios, requisitos de domínio e requisitos de partes interessadas/usuários normalmente ocorrem em um nível mais alto de abstração do que os requisitos do sistema. Quando um sistema consiste em uma hierarquia de subsistemas e componentes, temos requisitos de sistema nos níveis de abstração correspondentes para subsistemas e componentes.

Camadas típicas: negócios, sistema, componentes

Quando os requisitos de negócios e os requisitos das partes interessadas são expressos em produtos de trabalho duráveis - como especificações de requisitos de negócios, especificações de requisitos das partes interessadas ou documentos de visão - eles precedem a especificação dos requisitos do sistema. Por exemplo, em situações contratuais, onde um cliente solicita o desenvolvimento de um sistema de um fornecedor, o cliente frequentemente cria e libera uma especificação de requisitos das partes interessadas. O fornecedor então usa isso como base para produzir uma especificação de requisitos do sistema. Em outros projetos, os requisitos de negócios, os requisitos das partes interessadas e os requisitos do sistema podem evoluir juntos.

Dependências

Alguns produtos de trabalho, como requisitos individuais, esboços ou modelos de processo, ocorrem em todos os níveis. Outros produtos de trabalho são especificamente associados a determinados níveis. Por exemplo, uma especificação de requisitos do sistema está associada ao nível do sistema. Observe que um requisito individual em um alto nível de abstração pode ser refinado em vários requisitos detalhados em níveis mais concretos.

Escolhendo um nível de abstração adequado

A escolha do nível de abstração adequado depende particularmente do assunto a ser especificado e do objetivo da especificação. Por exemplo, se o assunto a ser especificado for uma parte de baixo nível do problema a ser resolvido, ele será especificado em um nível de abstração bastante baixo. É importante, no entanto, não misturar requisitos que estão em diferentes níveis de abstração. Por exemplo, na especificação de um sistema de informação de saúde, ao escrever um requisito detalhado sobre fotos em cartões de identificação do cliente, o parágrafo subsequente não deve declarar uma meta geral do sistema, como reduzir custos de saúde enquanto mantém o nível de serviço atual para os clientes. Em produtos de trabalho de pequeno e médio porte (por exemplo, histórias de usuários ou casos de uso), os requisitos devem estar mais ou menos no mesmo nível de abstração. Em grandes produtos de trabalho, como uma especificação de requisitos do sistema, os requisitos em diferentes níveis de abstração devem ser mantidos separados por meio da estruturação adequada da especificação (Seção 3.6).

Os requisitos ocorrem naturalmente em diferentes níveis de abstração. É útil selecionar produtos de trabalho adequados para um determinado nível de abstração e estruturar adequadamente produtos de trabalho que contenham requisitos em vários níveis de abstração.

3.1.3 Nível de Detalhe

Ao especificar os requisitos, os Engenheiros de Requisitos devem decidir sobre o nível de detalhe em que os requisitos devem ser especificados. No entanto, decidir qual nível de detalhe é apropriado ou mesmo ótimo para um determinado requisito é uma tarefa desafiadora.

Quantos detalhes?

Por exemplo, em uma situação em que o cliente e o fornecedor de um sistema colaboraram estreitamente, pode ser suficiente declarar um requisito sobre um formulário de entrada de dados da seguinte forma: "O sistema deve fornecer um formulário para inserir os dados pessoais do cliente." Por outro lado, em uma situação em que o projeto e a implementação do sistema são terceirizados para um fornecedor com pouco ou nenhum conhecimento do domínio, será necessária uma especificação detalhada do formulário de entrada do cliente.

O nível de detalhe para o qual os requisitos devem ser especificados depende de vários fatores, em particular:

Fatores que afetam o nível de detalhe necessário

- O problema e o contexto do projeto: quanto mais difícil for o problema e quanto menos familiarizados os Engenheiros de Requisitos e desenvolvedores estiverem com o contexto do projeto, mais detalhes serão necessários.
- O grau de compreensão compartilhada do problema: quando há baixa compreensão compartilhada implícita (ver Princípio 3 no Capítulo 2), são necessárias especificações explícitas e detalhadas para criar o grau necessário de compreensão compartilhada.
- O grau de liberdade deixado para designers e programadores: requisitos menos detalhados dão mais liberdade aos desenvolvedores.
- Disponibilidade de feedback rápido das partes interessadas durante o projeto e a implementação: quando o feedback rápido está disponível, especificações menos detalhadas são suficientes para controlar o risco de desenvolver o sistema errado.
- Custo x valor de uma especificação detalhada: quanto maior o benefício de um requisito, mais podemos nos dar ao luxo de especificá-lo em detalhes.
- Padrões e regulamentos: Padrões impostos e restrições regulatórias podem significar que os requisitos devem ser especificados com mais detalhes do que seria necessário.

Não existe um nível universalmente "certo" de detalhes para os requisitos. Para cada requisito, o nível adequado de detalhamento depende de muitos fatores. Quanto maior o nível de detalhamento dos requisitos especificados, menor o risco de eventualmente obter algo que tenha características ou propriedades inesperadas ou ausentes. No entanto, o custo da especificação aumenta à medida que o nível de detalhe aumenta.

3.1.4 Aspectos a Considerar

Independentemente dos produtos de trabalho de ER usados, vários aspectos precisam ser considerados ao especificar os requisitos [Glin2019].

Considerando vários aspectos

Primeiro, como existem requisitos funcionais, requisitos de qualidade e restrições (consulte a Seção 1.1), os Engenheiros de Requisitos devem certificar-se de que cobrem todos os três tipos de requisitos ao documentar os requisitos. Na prática, as partes interessadas tendem a omitir os requisitos de qualidade porque os consideram garantidos.

Eles também tendem a especificar restrições como requisitos funcionais. Portanto, é importante que os Engenheiros de Requisitos entendam isso corretamente.

Ao olhar para os requisitos funcionais, observamos que eles pertencem a diferentes aspectos, como, por exemplo, uma estrutura de dados necessária, uma ordem de ações necessária ou a reação necessária a algum evento externo. Distinguimos entre três aspectos principais: *estrutura e dados, função e fluxo e estado e comportamento*.

Aspectos dentro dos requisitos funcionais

O aspecto de *estrutura e dados* concentra-se nos requisitos relativos à estrutura estática de um sistema e aos dados (persistentes) que um sistema deve conhecer para executar as funções necessárias e fornecer os resultados necessários.

Estrutura e dados

O aspecto de *função e fluxo* lida com as funções que um sistema deve fornecer e o fluxo de controle e dados dentro e entre as funções para criar os resultados necessários de determinadas entradas.

Função e fluxo

O aspecto de *estado e comportamento* concentra-se na especificação do comportamento dependente do estado de um sistema - em particular, como um sistema deve reagir a qual evento externo, dependendo do estado atual do sistema.

Estado e comportamento

Ao lidar com *requisitos de qualidade*, como usabilidade, confiabilidade ou disponibilidade, um modelo de qualidade – por exemplo, o modelo fornecido pela ISO/IEC 25010 [ISO25010] – pode ser usado como uma lista de verificação.

Requerimentos de qualidade

Dentro dos requisitos de qualidade, os *requisitos de desempenho* são de particular importância. Os requisitos de desempenho tratam de:

- Tempo (por exemplo, para executar uma tarefa ou reagir a eventos externos)
- Volume (por exemplo, tamanho do banco de dados necessário)
- Frequência (por exemplo, de computar uma função ou receber estímulos de sensores)
- Taxa de transferência (por exemplo, transmissão de dados ou taxas de transação)
- Consumo de recursos (por exemplo, CPU, armazenamento, largura de banda, bateria)

Algumas pessoas também consideram a precisão necessária de um cálculo como um requisito de desempenho.

Sempre que possível, devem ser especificados valores mensuráveis. Quando os valores seguem uma distribuição de probabilidade, especificar apenas a média não é suficiente. Se a função de distribuição e seus parâmetros não puderem ser especificados, os Engenheiros de Requisitos devem se esforçar para especificar valores mínimos e máximos ou valores de 95% além das médias.

Documentar os requisitos de qualidade além dos requisitos de desempenho é notoriamente difícil.

Dificuldade de documentar os requisitos de qualidade

Representações qualitativas, como “O sistema deve ser seguro e fácil de usar”, são ambíguas e, portanto, difíceis de alcançar e validar.

As *representações quantitativas* são mensuráveis, o que é um grande trunfo em termos de atingir e validar sistematicamente um requisito de qualidade. No entanto, eles levantam dificuldades principais (por exemplo, como podemos afirmar a segurança em termos quantitativos?) e podem ser bastante caros para especificar.

As *representações operacionalizadas* declaram um requisito de qualidade em termos de requisitos funcionais para alcançar a qualidade desejada. Por exemplo, um requisito de segurança de dados pode ser expresso em termos de uma função de login que restringe o acesso aos dados e uma função que criptografa os dados armazenados. As representações operacionalizadas tornam os requisitos de qualidade testáveis, mas também podem implicar em decisões prematuras de projeto.

A regra frequentemente ouvida “Somente um requisito de qualidade quantificado é um requisito de boa qualidade” está ultrapassada e pode levar a requisitos de qualidade com valor baixo ou até negativo devido ao alto esforço envolvido na quantificação. Em vez disso, uma abordagem baseada em risco deve ser usada [Glin2008].

As representações qualitativas dos requisitos de qualidade são suficientes nas seguintes situações:

- Há compreensão compartilhada implícita suficiente entre as partes interessadas, engenheiros de requisitos e desenvolvedores.
- As partes interessadas, os engenheiros de requisitos e os desenvolvedores concordam com uma solução conhecida que satisfaça os requisitos.
- As partes interessadas só querem dar orientações gerais de qualidade e confiar nos desenvolvedores para obter os detalhes corretos.
- Circuitos curtos de feedback estão em vigor, de modo que os problemas possam ser detectados com antecedência.

Quando os desenvolvedores são *capazes de generalizar a partir de exemplos*, especificar os requisitos de qualidade em termos de exemplos quantificados ou comparações com um sistema existente é uma maneira barata e eficaz de documentar os requisitos de qualidade.

Apenas nos casos em que existe um *risco elevado* de não satisfação das necessidades das partes interessadas, nomeadamente quando os requisitos de qualidade são *críticos para a segurança*, deve ser considerada uma representação totalmente quantificada ou uma operacionalização em termos de requisitos funcionais.

Ao especificar *restrições*, as seguintes categorias de restrições devem ser consideradas:

[Restrições](#)

- *Técnico*: dadas interfaces ou protocolos, componentes ou frameworks que devem ser usados, etc.
- *Legal*: restrições impostas por leis, contratos, normas ou regulamentos *Organizacional*: pode haver restrições em termos de estruturas organizacionais, processos ou políticas que não devem ser alteradas pelo sistema.
- *Cultural*: os hábitos e expectativas do usuário são, até certo ponto, moldados pela cultura em que os usuários vivem. Este é um aspecto particularmente importante a ser considerado quando os usuários de um sistema vêm de culturas diferentes ou quando os Engenheiros de Requisitos e desenvolvedores estão enraizados em uma cultura diferente para os usuários do sistema.
- *Ambiental*: ao especificar sistemas ciberfísicos, condições ambientais como temperatura, umidade, radiação ou vibração podem ter que ser consideradas como restrições; consumo de energia e dissipação de calor podem constituir restrições adicionais.
- *Físico*: quando um sistema compreende componentes físicos ou interage com eles, o sistema torna-se limitado pelas leis da física e pelas propriedades dos materiais usados para os componentes físicos.
- Além disso, *soluções particulares ou restrições exigidas por partes interessadas importantes* também constituem restrições.

[Categorias de restrição a serem consideradas](#)

Por fim, os requisitos só podem ser entendidos no *contexto* (consulte o Princípio 4 no Capítulo 2). Consequentemente, um outro aspecto deve ser considerado, que chamamos de *contexto e limite*.

[Contexto e limite](#)

O aspecto *de contexto e fronteira* abrange requisitos de domínio e suposições de domínio no contexto do sistema, bem como os atores externos com os quais o sistema interage e as interfaces externas entre o sistema e seu ambiente na fronteira do sistema.

Existem muitas inter-relações e dependências entre os aspectos mencionados acima. Por exemplo, uma solicitação emitida por um usuário (contexto) pode ser recebida pelo sistema por meio de uma interface externa (fronteira), desencadear uma transição de estado do sistema (estado e comportamento), que inicia uma ação (função) seguida de outra ação (fluxo) que requer dados com alguma estrutura dada (estrutura e dados) para fornecer um resultado ao usuário (contexto) dentro de um determinado intervalo de tempo (qualidade).

Inter-relações e dependências entre aspectos

Alguns produtos de trabalho se concentram em um aspecto específico e se abstraem dos outros aspectos. Esse é particularmente o caso dos modelos de requisitos (Seção 3.4). Outros produtos de trabalho, como uma especificação de requisitos do sistema, cobrem todos esses aspectos. Quando aspectos diferentes são documentados em produtos de trabalho separados ou em capítulos separados do mesmo produto de trabalho, esses produtos de trabalho ou capítulos devem ser mantidos consistentes entre si.

Muitos aspectos diferentes precisam ser considerados ao documentar requisitos, em particular, funcionalidade (estrutura e dados, função e fluxo, estado e comportamento), qualidade, restrições e contexto circundante (contexto e limite).

3.1.5 Diretrizes Gerais de Documentação

Independentemente das técnicas utilizadas, existem algumas diretrizes gerais que devem ser seguidas na criação de produtos de trabalho de RE:

Diretrizes gerais

- Selecione um tipo de produto de trabalho adequado à finalidade pretendida.
- Evite a redundância referenciando o conteúdo em vez de repetir o mesmo conteúdo novamente.
- Evite inconsistências entre produtos de trabalho, principalmente quando abrangem aspectos diferentes.
- Use os termos de forma consistente, conforme definido no glossário.
- Estruture os produtos de trabalho adequadamente - por exemplo, usando padrões estruturais.

3.1.6 Planejamento do Produto de Trabalho

Cada configuração de projeto e cada domínio são diferentes, portanto, o conjunto de produtos de trabalho resultantes deve ser definido para cada empreendimento. As partes envolvidas, particularmente os Engenheiros de Requisitos, partes interessadas e proprietários ou gerentes de projetos/produtos precisam concordar com as seguintes questões:

O que considerar

- Em quais produtos de trabalho os requisitos devem ser registrados e para qual finalidade (ver Tabela 3.1)?
- Quais níveis de abstração precisam ser considerados (Seção 3.1.2)?
- Até que nível de detalhe os requisitos devem ser documentados em cada nível de abstração (Seção 0)?
- Como os requisitos devem ser representados nesses produtos de trabalho (por exemplo, baseados em linguagem natural ou baseados em modelo, veja abaixo) e quais notações devem ser usadas?

Os engenheiros de requisitos devem definir os produtos de trabalho de ER a serem usados em um estágio inicial de um projeto. Tal definição inicial:

Defina os produtos de trabalho de RE com antecedência

- Auxilia no planejamento de esforços e recursos
- Garante que as notações apropriadas sejam usadas
- Garante que todos os resultados sejam registrados nos produtos de trabalho corretos
- Garante que nenhuma grande reorganização de informações e “edição final” seja necessária
- Ajuda a evitar redundância, resultando em menos trabalho e facilidade de manutenção

3.2 Produtos de trabalho baseados em linguagem natural

A linguagem natural, tanto na forma falada quanto na escrita, sempre foi um meio central para comunicar os requisitos dos sistemas. Usar linguagem natural para escrever produtos de trabalho de RE tem muitas vantagens. Em particular, a linguagem natural é extremamente expressiva e flexível, o que significa que quase qualquer requisito concebível em qualquer aspecto pode ser expresso em linguagem natural. Além disso, a linguagem natural é usada na vida cotidiana e é ensinada na escola, portanto, não é necessário nenhum treinamento específico para ler e entender os requisitos escritos em linguagem natural.

Vantagens

A evolução humana moldou a linguagem natural como um meio de *comunicação falada entre pessoas que interagem diretamente*, onde mal-entendidos e informações ausentes podem ser detectados e corrigidos rapidamente. Portanto, a linguagem natural *não* é otimizada para comunicação precisa, inequívoca e abrangente por meio de documentos escritos. Isso constitui um grande problema ao escrever documentação técnica (como requisitos) em linguagem natural. Em contraste com a comunicação em linguagem natural *falada*, onde a comunicação é contextualizada e interativa com feedback imediato, não há meios naturais para detectar e corrigir rapidamente ambiguidades, omissões e inconsistências em textos escritos *em* linguagem natural. Pelo contrário, encontrar tais ambigüidades, omissões e inconsistências em textos escritos é difícil e caro, especialmente para produtos de trabalho que contêm uma grande quantidade de texto em linguagem natural.

problemas

O problema pode ser mitigado até certo ponto escrevendo documentação técnica conscientemente, seguindo regras comprovadas e evitando armadilhas conhecidas.

Regras de escrita

Ao escrever requisitos em linguagem natural, os Engenheiros de Requisitos podem evitar muitos mal-entendidos potenciais aplicando algumas regras simples:

- Escreva frases curtas e bem estruturadas. A regra geral é expressar um único requisito em uma frase em linguagem natural. Para obter uma boa estrutura, os Engenheiros de Requisitos devem usar modelos de frase (Seção 3.3). *Frases curtas*
- Crie produtos de trabalho bem estruturados. Além de escrever frases bem estruturadas (veja acima), os produtos de trabalho escritos em linguagem natural também devem ser bem estruturados como um todo. Uma maneira comprovada de fazer isso é usando uma estrutura hierárquica de partes, capítulos, seções e subseções, como geralmente é feito em livros técnicos. Os modelos de documento (Seção 3.3) ajudam você a obter uma boa estrutura. *Trabalho estruturado*
produtos

- ▶ Defina e use consistentemente uma terminologia uniforme. Criar e usar um glossário (Seção 3.5) é o meio principal para evitar mal-entendidos e inconsistências sobre a terminologia. *Terminologia uniforme*
- ▶ Evite usar termos e frases vagas ou ambíguas. *Indefinição, ambiguidade*
- ▶ Conheça e evite as armadilhas da redação técnica (veja abaixo). *Conhecendo as armadilhas*

Ao escrever documentos técnicos em linguagem natural, existem algumas armadilhas bem conhecidas que devem ser evitadas ou coisas que precisam ser usadas com cuidado (consulte, por exemplo, [GoRu2003]).

Os engenheiros de requisitos devem *evitar* escrever requisitos que contenham o seguinte:

- ▶ *Descrições incompletas.* Os verbos na linguagem natural geralmente vêm com um conjunto de espaços reservados para substantivos ou pronomes. Por exemplo, o verbo “dar” tem três espaços reservados para *quem dá* o *quê* para *quem*. Ao escrever um requisito em linguagem natural, todos os espaços reservados do verbo usado devem ser preenchidos. *Descrições incompletas*
- ▶ *Substantivos inespecíficos.* O uso de substantivos como “os dados” ou “o usuário” deixa muito espaço para diferentes interpretações por diferentes partes interessadas ou desenvolvedores. Eles devem ser substituídos por substantivos mais específicos ou ser mais específicos adicionando adjetivos ou atribuindo-lhes um tipo bem definido. *Substantivos inespecíficos*
- ▶ *Condições incompletas.* Ao descrever o que deve ser feito, muitas pessoas se concentram no caso normal, omitindo os casos excepcionais. Na redação técnica, essa é uma armadilha a ser evitada: quando algo acontece apenas se certas condições forem verdadeiras, essas condições devem ser declaradas, fornecendo cláusulas *então* e *senão*. *Condições incompletas*
- ▶ *Comparações incompletas.* Na comunicação falada, as pessoas tendem a usar comparativos (por exemplo, “o novo aplicativo de vídeo é muito melhor”) sem dizer com o que estão comparando, geralmente assumindo que isso está claro no contexto. Na redação técnica, as comparações devem incluir um objeto de referência, por exemplo, “mais rápido que 0,1 ms”. *Comparações incompletas*

Existem outras coisas que os Engenheiros de Requisitos precisam usar com cuidado, pois constituem armadilhas em potencial:

- ▶ *Voz passiva.* As frases na voz passiva não têm sujeito atuante. Se um requisito for declarado na voz passiva, isso pode ocultar quem é o responsável pela ação descrita no requisito, levando a uma descrição incompleta. *Cuidado*
- ▶ *Quantificadores universais.* Quantificadores universais são palavras como *todos*, *sempre* ou *nunca*, que são usadas para fazer afirmações que são universalmente verdadeiras. Em sistemas técnicos, entretanto, tais propriedades universais são raras. Sempre que os Engenheiros de Requisitos usam um quantificador universal, eles precisam refletir se estão declarando uma propriedade verdadeiramente universal ou se estão especificando uma regra geral que tem exceções (que também precisam especificar). Eles devem ter o mesmo cuidado ao usar cláusulas “ou-ou” que, por sua semântica, excluem quaisquer outros casos excepcionais. *Voz passiva*
- ▶ *quantificadores universais*

Nominalizações. Quando um substantivo é derivado de um verbo (por exemplo, “autenticação” de “autenticar”), os linguistas chamam isso de nominalização. Ao especificar requisitos, os Engenheiros de Requisitos precisam lidar com as nominalizações com cuidado porque uma nominalização pode ocultar requisitos não especificados. Por exemplo, o requisito “Somente após autenticação bem-sucedida, o sistema deve fornecer acesso de usuário a (...)" implica que existe um procedimento para autenticação de usuários. Ao redigir tal requisito, portanto, o Engenheiro de Requisitos deve verificar se também há requisitos sobre o procedimento para autenticação de usuários legítimos.

Nominalizações

A linguagem natural é um meio muito poderoso para escrever requisitos. Para atenuar as desvantagens inerentes ao uso de linguagem natural para documentação técnica, os Engenheiros de Requisitos devem seguir regras de redação comprovadas e evitar armadilhas conhecidas.

3.3 Produtos de Trabalho Baseados em Modelos

Conforme mencionado na Seção 3.2 acima, o uso de modelos é um meio comprovado de escrever produtos de trabalho bons e bem estruturados em linguagem natural e, assim, atenuar alguns dos pontos fracos da linguagem natural para redação técnica. Um modelo é uma espécie de projeto pronto para a estrutura sintática de um produto de trabalho. Ao usar linguagem natural em RE, distinguimos entre três classes de modelos: modelos de frase, modelos de formulário e modelos de documento.

3.3.1 Modelos de Frases

Definição 3.2. Modelo de frase: um modelo para a estrutura sintática de uma frase que expressa um requisito individual ou uma história de usuário em linguagem natural.

modelo de frase

Um modelo de frase fornece uma estrutura de esqueleto com espaços reservados, nos quais os Engenheiros de Requisitos preenchem os espaços reservados para obter sentenças uniformes e bem estruturadas que expressam os requisitos.

O uso de modelos de frase é uma prática recomendada ao escrever requisitos individuais em linguagem natural e ao escrever histórias de usuários.

3.3.1.1 Modelos de Frases para Requisitos Individuais

Vários modelos de frase para escrever requisitos individuais foram definidos, por exemplo, em [ISO29148], [MWHN2009] e [Rupp2014]. A norma ISO/IEC/IEEE 29148 [ISO29148] fornece um modelo único e uniforme para requisitos individuais como segue:

*Modelo de frase
ISO/IEC/IEEE 29148*

[<Condição>] <Assunto> <Ação> <Objetos> [<Restrição>].

Exemplo: Quando um cartão válido é detectado, o sistema deve exibir a mensagem “Insira seu PIN” na tela de diálogo em 200 ms.

Ao formular uma ação com este modelo, as seguintes convenções sobre o uso de verbos auxiliares são freqüentemente usadas na prática:

Usando verbos auxiliares

- Deve denotar um requisito obrigatório.
- Deve denotar um requisito que não é obrigatório, mas fortemente desejado.
- *May* denota uma sugestão.
- *Will* (ou usando um verbo no tempo presente sem um dos verbos auxiliares mencionados acima) denota uma declaração factual que não é considerada um requisito.

Quando não há significados acordados para verbos auxiliares em um projeto, ou em caso de dúvida, definições como as dadas acima devem fazer parte de uma especificação de requisitos.

EARS (Easy Approach to Requirements Syntax) [MWHN2009] fornece um conjunto de modelos de frases que são adaptados a diferentes situações, conforme descrito abaixo. Requisitos ubíquos (devem ser sempre válidos):

modelos EARS

O <nome do sistema> deve <resposta do sistema>.

Requisitos orientados a eventos (acionados por um evento externo):

QUANDO <pré-condições opcionais> <gatilho> o <nome do sistema> deve
<resposta do sistema>.

Comportamento indesejado (descrevendo situações a serem evitadas):

SE <pré-condições opcionais> <gatilho>, ENTÃO o <nome do sistema> deve
<resposta do sistema>.

Observação: embora o modelo de comportamento indesejado seja semelhante ao modelo orientado a eventos, Mavin et al. fornecem um modelo separado para o último, argumentando que o comportamento indesejado (principalmente devido a eventos inesperados no contexto, como falhas, ataques ou coisas nas quais ninguém pensou) é uma importante fonte de omissões em RE.

Requisitos orientados pelo estado (aplicam-se apenas em determinados estados):

ENQUANTO <em um estado específico> o <nome do sistema> deve <resposta do sistema>.

Recursos opcionais (aplicáveis apenas se algum recurso estiver incluído no sistema):

ONDE <recurso está incluído> o <nome do sistema> deve <resposta do sistema>.

Na prática, frases que combinam as palavras-chave WHEN, WHILE e WHERE podem ser necessárias para expressar requisitos complexos.

O EARS foi projetado principalmente para a especificação de sistemas ciberfísicos. No entanto, também pode ser adaptado para outros tipos de sistemas.

3.3.1.2 Modelos de Frases para Histórias de Usuário

O modelo de frase clássico para escrever histórias de usuários foi introduzido por Cohn [Cohn2004]:

Modelo de história do usuário de Cohn

Como um <papel> eu quero <requisito> para que <benefício>.

Exemplo: “Como gerente de linha, desejo fazer consultas ad hoc ao sistema de contabilidade para que eu possa fazer o planejamento financeiro do meu departamento.”

Embora Cohn tenha designado a parte <benefício> do modelo como opcional, hoje em dia é uma prática padrão especificar um benefício para cada história de usuário.

Cada história de usuário deve ser acompanhada por um conjunto de *critérios de aceitação*, ou seja, critérios que a implementação da história de usuário deve satisfazer para ser aceita pelos interessados. Os critérios de aceitação tornam uma história de usuário mais concreta e menos ambígua. Isso ajuda a evitar erros de implementação devido a mal-entendidos.

Critérios de aceitação

3.3.2 Modelos de formulário

Definição 3.3. Modelo de formulário: Um modelo que fornece um formulário com campos predefinidos a serem preenchidos.

modelo de formulário

Os modelos de formulário são usados para estruturar produtos de trabalho de tamanho médio, como casos de uso. Cockburn [Cock2001] apresentou um modelo de formulário popular para casos de uso. [Laue2002] propôs um modelo para descrições de tarefas. A Tabela 3.2 mostra um modelo de formulário simples para casos de uso. Cada etapa do fluxo pode ser subdividida em uma ação de um ator e a resposta do sistema.

Tabela 3.2 Um modelo de formulário simples para escrever casos de uso

Nome	<Uma frase verbal ativa curta>
Condição prévia	<Condição(ões) que deve(m) ser mantida(s) quando a execução do caso de uso é acionada>
Condição final de sucesso	<Declarar após a conclusão bem-sucedida do caso de uso>
Condição final com falha	<Estado após falha na execução do caso de uso>
ator principal	<Nome do ator>
Outros atores	<Lista de outros atores envolvidos, se houver>
Acionar	<Evento que inicia a execução do caso de uso>
fluxo normal	<Descrição do cenário de sucesso principal em uma sequência de etapas: <ação 1> <passo 1> <ação 2> <passo 2> ... <etapa n> <ação n> ... >
fluxos alternativos	<Descrição das etapas alternativas ou excepcionais, com referência às etapas correspondentes no fluxo normal> <Extensões ao fluxo normal (se houver), com referência
Extensões	<às etapas estendidas no fluxo normal> <Campo opcional para maiores informações, como desempenho, frequência, relação com outros casos de uso, etc.>
Informação relacionada	

Modelo de caso de uso

Os modelos de formulário também são úteis para escrever requisitos de qualidade de forma mensurável [Gilb1988]. A Tabela 3.3 fornece um modelo de formulário simples para requisitos de qualidade mensuráveis, juntamente com um exemplo.

Tabela 3.3 Um modelo de formulário para especificar requisitos de qualidade mensuráveis

Modelo	Exemplo	<i>Modelo de requisitos de qualidade mensuráveis</i>
EU IA	<Número de requisitos>	R137,2
Meta	<Meta declarada qualitativamente>	Confirme as reservas de quarto imediatamente
Escala	<Escala para medir o requisito>	Tempo decorrido em segundos (escala de razão)
Metro	<Procedimento para medir o requisito>	Marcar os momentos em que o usuário pressiona o botão “Reservar” e quando o aplicativo exibe a confirmação. Medindo a diferença de tempo.
Mínimo	<Qualidade mínima aceitável a ser alcançada>	Menos de 5 s em pelo menos 95% de todos os casos
Faixa OK	<Intervalo de valores que está OK e visa>	Entre 0,5 e 3 s em mais de 98% de todos os casos
desejado	<Qualidade alcançada no melhor caso possível>	Menos de 0,5 s em 100% de todos os casos

Modelo de requisitos de qualidade mensuráveis

3.3.3 Modelos de Documentos

Definição 3.4. Modelo de documento: Um modelo que fornece uma estrutura de esqueleto predefinida para um documento.

modelo de documento

Os modelos de documento ajudam a estruturar sistematicamente os documentos de requisitos, por exemplo, uma especificação de requisitos do sistema. Modelos de documentos RE podem ser encontrados em normas, por exemplo, em [ISO29148]. O modelo Volere de Robertson e Robertson [RoRo2012], [Vole2020] também é popular na prática. Quando uma especificação de requisitos é incluída no conjunto de produtos de trabalho que um cliente solicitou e pagará, esse cliente pode prescrever o uso de modelos de documentos fornecidos por

o cliente. Na Figura 3.1, mostramos um exemplo de um modelo de documento simples para uma especificação de requisitos de sistema.

3.3.4 Vantagens e Desvantagens

O uso de modelos ao escrever produtos de trabalho de RE em linguagem natural tem grandes vantagens. Os modelos fornecem uma estrutura clara e reutilizável para produtos de trabalho, fazem com que pareçam uniformes e, assim, melhoram a legibilidade dos produtos de trabalho. Os modelos também ajudam você a capturar as informações mais relevantes e a cometer menos erros de omissão. Por outro lado, existe uma armadilha potencial quando os Engenheiros de Requisitos usam modelos mecanicamente, focando na estrutura sintática e não no conteúdo, negligenciando tudo o que não se encaixa no modelo.

<p>Parte I: Introdução</p> <ul style="list-style-type: none"> 1. Propósito do sistema 2. Escopo do desenvolvimento do sistema 3. Partes interessadas <p>Parte II: visão geral do sistema</p> <ul style="list-style-type: none"> 4. Visão e objetivos do sistema 5. Contexto e limites do sistema 6. Estrutura geral do sistema 7. Características do usuário <p>Parte III: Requisitos do sistema</p> <p>Organizado hierarquicamente de acordo com a estrutura do sistema, usando um esquema de numeração hierárquica para requisitos</p> <p>Por subsistema/componente:</p> <ul style="list-style-type: none"> • Requisitos funcionais (estrutura e dados, função e fluxo, estado e comportamento) • Requerimentos de qualidade • Restrições • Interfaces <p>Referências</p> <p>Apêndices</p> <p>Glossário (se não for gerenciado como um produto de trabalho próprio)</p> <p>Suposições e dependências</p>	<i>Exemplo de modelo de documento</i>
--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	---------------------------------------

Figura 3.1 Um modelo simples de especificação de requisitos de sistema

O uso de modelos ao escrever produtos de trabalho de RE em linguagem natural melhora a qualidade dos produtos de trabalho, desde que os modelos não sejam mal utilizados apenas como um exercício sintático.

3.4 Produtos de trabalho baseados em modelo

Requisitos formulados em linguagem natural podem ser facilmente lidos por pessoas, desde que falem o idioma. A linguagem natural sofre de ambigüidade devido à imprecisão da semântica de palavras, frases e sentenças [Davi1993]. Essa imprecisão pode levar a confusão e omissões nos requisitos. Ao ler os requisitos textuais, você tentará interpretá-los à sua maneira. Muitas vezes tentamos imaginar esses requisitos em nossa mente. Quando o número de requisitos é

Os requisitos em linguagem natural têm suas limitações

gerenciável, é possível manter uma visão e uma visão geral dos requisitos textuais. Quando o número de requisitos textuais se torna “muito grande”, perdemos a visão geral. Esse limite é diferente para cada pessoa. O número de requisitos textuais não é a única razão para perder visão e visão geral. A complexidade dos requisitos, o relacionamento entre os requisitos e a abstração dos requisitos também contribuem para isso. Você pode ter que ler os requisitos formulados em linguagem natural várias vezes antes de obter uma imagem correta e completa que o sistema deve cumprir. Temos uma capacidade limitada de processar requisitos em linguagem natural.

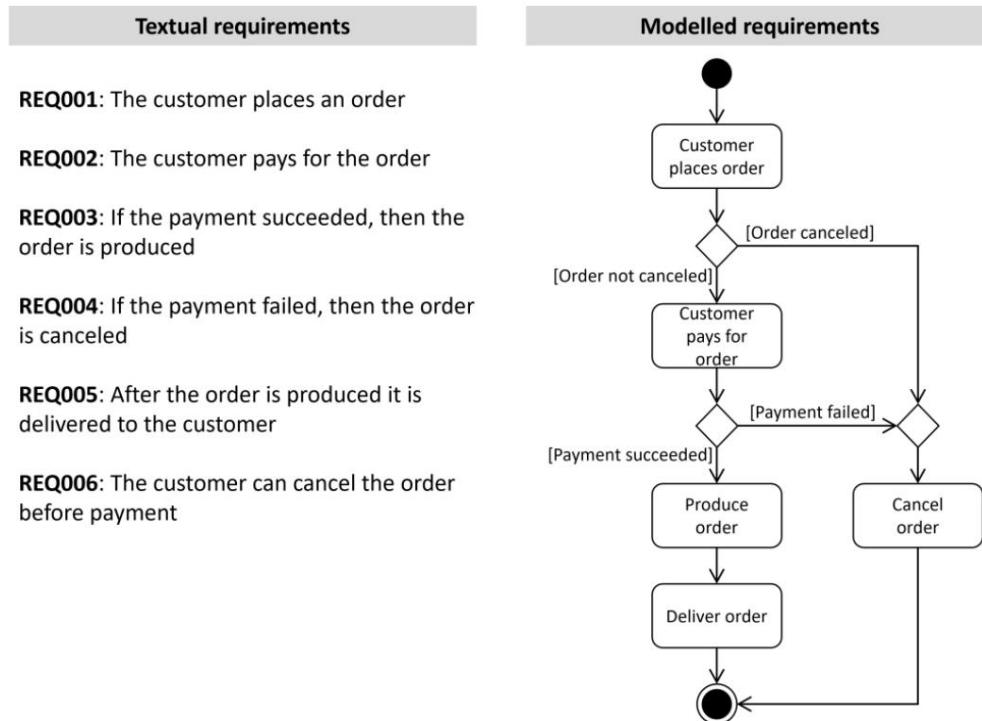


Figura 3.2 Requisitos textuais versus requisitos modelados

Um modelo é uma representação abstrata de uma parte existente da realidade ou de uma parte da realidade a ser criada. Exibir os requisitos (também) com um modelo (ou imagem) contribuirá para que os leitores compreendam os requisitos. Essa representação esquemática de um modelo é chamada de diagrama.

O diagrama na Figura 3.2 mostra rapidamente o que o sistema deve fornecer, mas apenas se você dominar a linguagem de modelagem. É evidente que se você não entender o diagrama, neste caso um diagrama de atividades UML, a imagem não contribuirá para um melhor entendimento dos requisitos.

Na próxima seção (3.4.1), o conceito de um modelo de requisitos é explicado. A modelagem de requisitos e objetivos de negócios é explicada na Seção 3.4.6. Um método importante para descrever a demarcação de um sistema é o modelo de contexto. Exemplos do contexto são descritos na Seção 3.4.2. As Seções 3.4.3 a 3.4.5 fornecem vários exemplos de linguagens de modelagem que são frequentemente usadas na prática da engenharia de sistemas.

Um modelo é uma representação abstrata da realidade

Os requisitos de modelagem contribuem para manter uma visão geral e uma visão dos requisitos

3.4.1 O Papel dos Modelos na Engenharia de Requisitos

Como qualquer linguagem, uma linguagem de modelagem consiste em regras gramaticais e uma descrição do significado das construções da linguagem, consulte a Seção 3.4.1.1. Embora um

modelo é uma representação visual da realidade, as regras de linguagem são importantes para entender o modelo e as nuances do modelo.

Nem sempre é eficiente ou eficaz resumir os requisitos em um modelo. Compreendendo as propriedades de um modelo, podemos determinar melhor quando podemos aplicar qual modelo, consulte a Seção 3.4.1.2.

Assim como a linguagem natural tem vantagens e desvantagens para expressar os requisitos, os modelos também têm. Se observarmos esses fatos ao aplicar um modelo, poderemos determinar melhor o valor agregado de aplicar o modelo "correto". Isso é discutido na Seção 3.4.1.3.

Muitos modelos já foram padronizados e são utilizados em diversos campos de aplicação, ver Seção 3.4.1.4. Considere, por exemplo, a construção de uma casa, onde um arquiteto usa um modelo padronizado para descrever a casa. Um exemplo de modelos usados por arquitetos de construção são os modelos de informações de construção (BIM) [ISO19650], que modelam os elementos necessários para planejar, construir e gerenciar edifícios e outros elementos de construção.

Outro exemplo é a eletrônica, onde o desenho de diagramas eletrônicos é padronizado para que os profissionais possam entender, calcular e realizar a eletrônica.

Para determinar se um diagrama é aplicado corretamente, podemos validar os critérios de qualidade de um diagrama. Esses critérios são descritos na Seção 3.4.1.5.

3.4.1.1 Sintaxe e Semântica

Se você pensar em uma linguagem natural, por exemplo, sua língua nativa, ela é definida por sua gramática e semântica.

A gramática descreve os elementos (palavras e frases) e as regras que a língua deve obedecer. Em uma linguagem de modelagem, isso é chamado de sintaxe, veja a Figura 3.3.

A sintaxe descreve quais elementos de notação (símbolos) são usados na linguagem. Ele também descreve como esses elementos de notação podem ser usados em combinação.

Uma linguagem de modelagem consiste em sintaxe e semântica

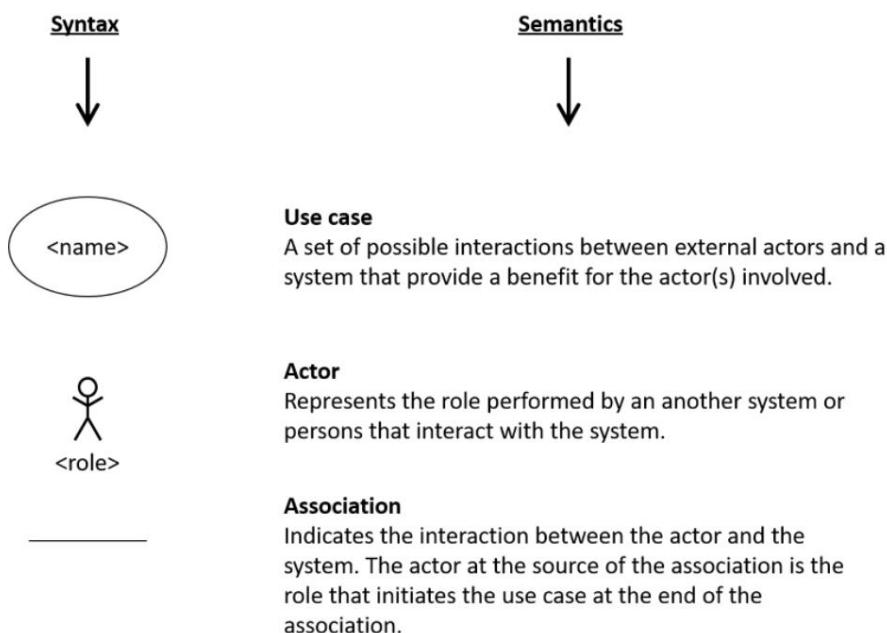


Figura 3.3 Sintaxe e semântica da linguagem de modelagem

A semântica define o significado dos elementos de notação e define o significado da combinação de elementos. Entender o significado dos elementos da notação é fundamental para evitar o risco de o modelo ser mal interpretado.

3.4.1.2 Propriedades de um Modelo

Um modelo de requisitos é um modelo conceitual que descreve os requisitos para o sistema a ser desenvolvido. Um modelo também é usado para representar a situação atual para entender, analisar e explorar os problemas atuais. Nesse contexto, conceitual significa que a realidade é reduzida à sua essência. Um modelo tem um alto nível de abstração e reduz a realidade ao que é relevante nesse nível genérico.

Uma linguagem de modelagem conceitual pode ser padronizada (internacionalmente) e é então referida como uma linguagem de modelagem formal. Um exemplo disso é a linguagem de modelagem amplamente utilizada e frequentemente aplicada UML (Unified Modeling Language).

Um modelo tem várias propriedades que são exploradas nas seções a seguir:

- Um modelo é feito para um propósito específico.
- Um modelo dá uma representação da realidade.
- Um modelo é usado para reduzir informações para que possamos entender melhor realidade ou focar em parte da realidade.

Um modelo é uma representação abstrata de uma parte existente da realidade ou de uma parte da realidade a ser criada. A noção de realidade inclui qualquer conjunto concebível de elementos, fenômenos ou conceitos, incluindo outros modelos. A parte modelada da realidade é chamada de original. O processo para descrever o original pode ser descritivo ou prescritivo.

Um modelo é uma representação abstrata de seu original na realidade

Modelar o original existente é chamado de modelagem descritiva. Ele mostra a realidade atual e reflete os requisitos que são atendidos. Se ainda não existe um modelo do original, esse modelo é o resultado da análise da situação atual.

A modelagem da realidade é feita de forma descritiva ou prescritiva

Modelar um original a ser criado é chamado de modelagem prescritiva. Indica qual realidade futura é esperada ou necessária. Se existe um modelo com propriedades descritivas para a situação dada, então um modelo com propriedades prescritivas pode ser derivado do original, indicando quais requisitos serão novos, alterados ou não são mais necessários. O modelo prescritivo descreve a situação futura final desejada.

A realidade pode ser complexa. Se aplicarmos detalhes “demais”, um modelo pode ser difícil de entender. Essa realidade complexa pode ser simplificada reduzindo a quantidade de informações no modelo. Em um modelo, podemos omitir informações irrelevantes. Reduzir a quantidade de informação pode nos dar uma melhor compreensão da realidade e nos permitir entender a essência dessa realidade com mais facilidade. Com base na finalidade pretendida (primeira propriedade) para a qual o modelo é aplicado, apenas as informações relevantes são exibidas no modelo.

Um modelo reduz Informação

Observe que, se a informação “demais” for reduzida, pode surgir uma imagem nublada ou incorreta da realidade. Portanto, deve-se considerar cuidadosamente quanto da informação pode ser reduzida sem distorcer a realidade.

Existem várias maneiras de reduzir as informações:

- Por compactação ou agregação

Aregar informações é uma forma de tornar as informações mais abstratas. A informação é despojada de detalhes irrelevantes e, portanto, mais compacta.

A informação é, por assim dizer, condensada.

▶ Por seleção

Ao selecionar apenas as informações relevantes, e não todas, é possível indicar qual é o assunto em questão. O foco está em uma parte específica ou número de partes do total.

Ambas as formas de redução de informações também podem ser aplicadas em conjunto.

Um modelo é uma representação da realidade e cada modelo representa certos aspectos da realidade. Por exemplo, um desenho de construção mostra a divisão do espaço em um edifício e um diagrama elétrico mostra a fiação do circuito elétrico.

Um modelo se adapta a um determinado propósito

Ambos os modelos representam o edifício para uma finalidade específica. Um modelo é feito para um propósito específico em um contexto específico. No exemplo acima, o contexto é o projeto e/ou realização de um edifício. Os vários desenhos de construção representam informações sobre um aspecto específico do edifício. Isso torna imediatamente claro que um modelo específico só pode ser usado se se adequar ao propósito para o qual o modelo foi feito.

3.4.1.3 Vantagens e Desvantagens dos Requisitos de Modelagem

Em comparação com as línguas naturais, os modelos apresentam as seguintes vantagens, entre outras:

Vantagens dos modelos

- ▶ Os elementos e suas conexões são mais fáceis de entender e lembrar.

Uma imagem fala mais que mil palavras. Uma imagem, e também um modelo, pode ser mais fácil de entender e lembrar. Observe que um modelo não é autoexplicativo e precisa de informações extras – ou seja, uma legenda, exemplos, cenários, etc.

- ▶ O foco em um único aspecto reduz a carga cognitiva necessária para entender os requisitos modelados.

Como um modelo tem um propósito específico e uma quantidade reduzida de informações, entender a realidade modelada pode exigir menos esforço.

- ▶ As linguagens de modelagem de requisitos possuem uma sintaxe restrita que reduz possíveis ambiguidades e omissões.

Como a linguagem de modelagem (sintaxe e semântica) é mais simples, ou seja, número limitado de elementos de notação e regras de linguagem mais rígidas em comparação com a linguagem natural, o risco de confusão e omissões é menor.

- ▶ Maior potencial para análise automatizada e processamento de requisitos.

Como uma linguagem de modelagem é mais formal (número limitado de elementos de notação e regras de linguagem mais rígidas) do que uma linguagem natural, ela se presta melhor para automatizar a análise ou processamento de requisitos.

Apesar das grandes vantagens na visualização de requisitos com modelos, os modelos também possuem suas limitações.

Desvantagens dos modelos

- ▶ Manter modelos que enfocam diferentes aspectos consistentes entre si é um desafio.

Se vários modelos forem usados para descrever os requisitos, é importante manter esses modelos consistentes entre si. Isso requer muita disciplina e coordenação entre os modelos.

- Informações de diferentes modelos precisam ser integradas para compreensão causal.

Se vários modelos forem usados, todos os modelos devem ser compreendidos para permitir um bom entendimento dos requisitos.

- Os modelos se concentram principalmente nos requisitos funcionais.

Os modelos para descrever requisitos e restrições de qualidade são limitados, se não carentes de contexto específico. Esses tipos de requisitos devem ser fornecidos em linguagem natural junto com os modelos, por exemplo, como um produto de trabalho separado.

- A sintaxe restrita de uma linguagem de modelagem gráfica implica que nem todo item relevante de informação pode ser expresso em um modelo.

Como um modelo é feito para um propósito e contexto específicos, nem sempre é possível registrar todos os requisitos no modelo ou em vários modelos.

Os requisitos que não podem ser expressos em modelos são adicionados ao modelo como requisitos de linguagem natural ou como um produto de trabalho separado.

Portanto, os modelos de requisitos devem ser sempre acompanhados de linguagem natural [Davi1995].

3.4.1.4 Aplicação dos Modelos de Requisitos

Conforme indicado nas seções anteriores, existem modelos comuns para vários contextos.

Por exemplo, na arquitetura, você tem desenhos de construção, diagramas de tubulação, diagramas elétricos, etc. para expressar as especificações de um edifício. Em outros contextos - por exemplo, desenvolvimento de software - existem linguagens de modelagem que são úteis nesses tipos de contexto. Um aspecto importante na aplicação de modelos é utilizar modelos que sejam comuns no contexto ou que tenham sido desenvolvidos especialmente para um determinado contexto.

Muitas linguagens de modelagem — por exemplo, UML [OMG2017] ou BPMN [OMG2013] — foram padronizadas. Quando os requisitos são especificados em uma linguagem de modelagem não padrão, a sintaxe e a semântica da linguagem devem ser explicadas ao leitor, por exemplo, por meio de uma legenda.

Modelos são usados para descrever os requisitos de uma certa perspectiva. No desenvolvimento do sistema, os requisitos funcionais são categorizados nas seguintes perspectivas (consulte também a Seção 3.1.4):

- Estrutura e dados

Modelos que se concentram nas propriedades estruturais estáticas de um sistema ou domínio

- Função e fluxo

Modelos que se concentram na sequência de ações necessárias para produzir os resultados necessários a partir de determinadas entradas ou ações necessárias para executar um processo (de negócios), incluindo o fluxo de controle e dados entre as ações e quem é responsável por qual ação

- Estado e comportamento

Modelos que se concentram no comportamento de um sistema ou no ciclo de vida de objetos de negócios em termos de reações dependentes do estado a eventos ou na dinâmica da interação de componentes

A natureza do sistema que está sendo modificado ou construído dá direção aos modelos a serem usados. Por exemplo, se a natureza do sistema é processar informações e relacionamentos, espera-se que haja muitos requisitos funcionais que descrevam essas informações e esses relacionamentos. Como resultado, usamos uma linguagem de modelagem correspondente que se presta à modelagem de dados e sua estrutura.

A natureza de um sistema ajuda na seleção do modelo apropriado

Naturalmente, um sistema consistirá em uma combinação das perspectivas acima. Segue-se que um sistema precisa ser modelado a partir de múltiplas perspectivas. As seções 3.4.3 a 3.4.5 detalham os diferentes modelos para cada perspectiva.

Antes que os requisitos sejam levantados e documentados - por exemplo, com modelos - é feito um inventário dos objetivos e do contexto. Estes também podem ser modelados, veja Seções 3.4.6 respectivamente 3.4.2.

A aplicação de modelos nos ajuda principalmente das seguintes maneiras:

- *Especificando* requisitos (principalmente funcionais) em parte ou mesmo completamente, como um meio de substituir os requisitos representados textualmente
- *Decompondo* uma realidade complexa em aspectos bem definidos e complementares; sendo cada aspecto representado por um modelo específico, ajudando-nos a compreender a complexidade da realidade
- *Parafrasear* requisitos representados textualmente para melhorar a sua compreensão, em particular no que diz respeito às relações entre eles
- *Validar* requisitos representados textualmente com o objetivo de descobrir omissões, ambiguidades e inconsistências

A modelagem dos requisitos também ajuda na estruturação e análise do conhecimento. Você pode usar diagramas para estruturar seus próprios pensamentos para obter uma melhor compreensão do sistema e seu contexto.

3.4.1.5 Aspectos de Qualidade de um Modelo de Requisitos

Esta é uma seção suplementar para a qual não haverá perguntas no exame de nível CPRE Foundation.

Uma parte substancial dos modelos de requisitos são diagramas ou representações gráficas. A qualidade do modelo de requisitos é determinada pela qualidade dos diagramas individuais e seus relacionamentos mútuos. Por sua vez, a qualidade dos diagramas individuais é determinada pela qualidade dos elementos do modelo dentro dos diagramas. A qualidade dos modelos de requisitos e elementos do modelo pode ser avaliada em três critérios [LiSS1994]:

A qualidade de um modelo é determinada por três critérios

- qualidade sintática
- qualidade semântica
- qualidade pragmática

A qualidade sintática expressa até que ponto um único elemento do modelo (gráfico ou textual), diagrama de requisitos ou modelo de requisitos atende às especificações sintáticas. Se, por exemplo, um modelo que descreve os requisitos como um modelo de classe contém elementos de modelagem que não fazem parte da sintaxe ou os elementos do modelo são mal utilizados, isso diminuirá a qualidade sintática do modelo. Uma parte interessada desse modelo - por exemplo, um testador - pode interpretar mal as informações representadas pelo modelo. Isso pode eventualmente levar a casos de teste inadequados.

qualidade sintática

As ferramentas de modelagem de requisitos fornecem facilidades para verificar a qualidade sintática dos modelos.

A qualidade semântica expressa até que ponto um único elemento do modelo (gráfico ou textual), o diagrama de requisitos ou o modelo de requisitos representa correta e completamente os fatos.

qualidade semântica

Assim como na linguagem natural, a semântica dá sentido às palavras. Se um termo pode ter significados diferentes ou existem vários termos que significam a mesma coisa, isso pode levar a falhas de comunicação. O mesmo se aplica à semântica dos elementos de modelagem. Se os elementos de modelagem forem mal interpretados ou aplicados incorretamente, o modelo pode ser mal interpretado.

A qualidade pragmática expressa até que ponto um único elemento do modelo (gráfico ou textual), o diagrama de requisitos ou o modelo de requisitos é adequado para o uso pretendido - ou seja, se o grau de detalhe e o nível de abstração são apropriados para o uso pretendido e se o modelo apropriado é selecionado em relação ao domínio ou contexto. Isso pode ser avaliado se o propósito e as partes interessadas do diagrama forem conhecidos. Versões intermediárias do modelo podem ser submetidas às partes interessadas para validar se os diagramas atendem ao seu propósito.

qualidade pragmática

Durante a validação dos requisitos, a qualidade dos diagramas de modelagem usados é avaliada para garantir que esses diagramas atendam ao propósito e à utilidade pretendidos.

3.4.1.6 Melhor dos dois mundos

Conforme explicado na seção anterior, os requisitos expressos em forma textual ou visual/gráfica (ou seja, por meio de modelos de requisitos) têm suas vantagens e desvantagens. Ao usar representações textuais e gráficas dos requisitos, podemos aproveitar o poder e os benefícios de ambas as formas de representação.

Corrigir um modelo com requisitos textuais adiciona mais significado ao modelo. Outra combinação útil é que podemos vincular requisitos e restrições de qualidade a um modelo ou elemento de modelagem específico. Isso fornece uma imagem mais completa dos requisitos específicos.

Requisitos de documento em linguagem natural e modelos para se beneficiar dos pontos fortes de ambas as abordagens

O uso de modelos também pode dar suporte aos requisitos textuais. A adição de modelos e imagens aos requisitos textuais suporta esses modelos para uma melhor compreensão e visão geral.

3.4.2 Contexto do Sistema de Modelagem

O Capítulo 2, Princípio 4 introduz a noção de que os requisitos nunca vêm isoladamente e que o contexto do sistema, como sistemas, processos e usuários existentes, precisa ser considerado ao definir os requisitos para o sistema novo ou alterado.

Os modelos de contexto especificam a incorporação estrutural do sistema em seu ambiente, com suas interações com os usuários do sistema, bem como com outros sistemas novos ou existentes dentro do contexto relevante. Um modelo de contexto não é uma descrição gráfica dos requisitos, mas é usado para revelar algumas das fontes dos requisitos. A Figura 3.4 fornece um exemplo abstrato de um sistema e seu ambiente, com suas interfaces para os usuários do sistema e suas interfaces para outros sistemas. Assim, os diagramas de contexto ajudam a identificar as interfaces do usuário, bem como as interfaces do sistema. Se o sistema interagir com os usuários, as interfaces do usuário devem ser especificadas em uma etapa posterior durante o ER. Se o sistema interage com outros sistemas, as interfaces para esses sistemas devem ser definidas de forma mais

Os modelos de contexto ajudam a entender o contexto e os limites de um sistema

detalhes em uma etapa posterior. Interfaces para outros sistemas podem já existir ou podem precisar ser desenvolvidas ou modificadas.

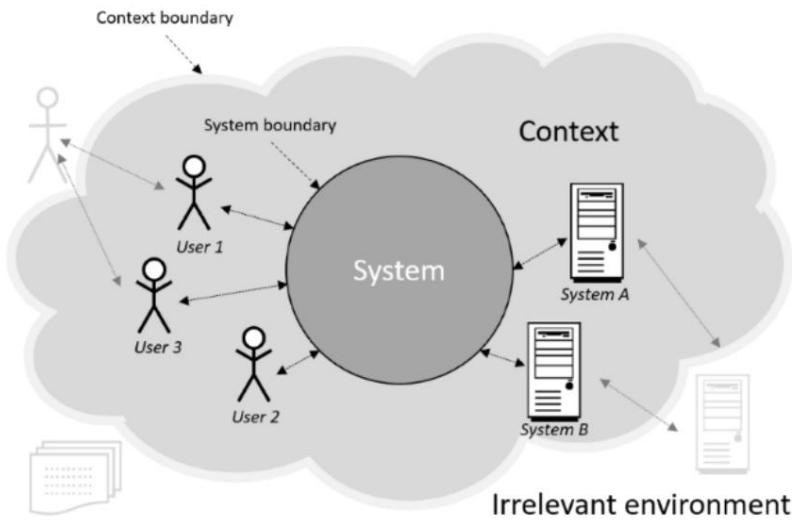


Figura 3.4 Um sistema em seu contexto

Mesmo que não haja uma linguagem de modelagem padronizada para modelos de contexto, os modelos de contexto são frequentemente representados por:

- Diagramas de fluxo de dados de análise estruturada [DeMa1978]
- Diagramas de caso de uso UML [OMG2017]

Observação: o modelo de caso de uso UML consiste em dois elementos; o diagrama de caso de uso da UML (consulte a Figura 3.6) e a especificação do caso de uso (Seção 3.4.2.2). Este capítulo enfoca a modelagem com os diagramas de caso de uso UML.
- Diagramas de caixa e linha personalizados [Glin2019]

No domínio da engenharia de sistemas, os diagramas de definição de blocos SysML [OMG2018] podem ser adaptados para expressar modelos de contexto usando blocos estereotipados para o sistema e os atores.

Nas próximas duas subseções, apresentamos a notação de diagramas de fluxo de dados (DFD) e diagramas de casos de uso UML para modelar o contexto de um sistema. Esses dois exemplos não descrevem o contexto completo, mas enfatizam o contexto de um ponto de vista específico.

3.4.2.1 Diagrama de Fluxo de Dados

O contexto do sistema pode ser visto de diferentes perspectivas. A análise estruturada de sistemas [DeMa1978] fala sobre o diagrama de contexto. Este diagrama é um diagrama de fluxo de dados (DFD) especial onde o sistema é representado por um processo (o sistema). A Figura 3.5 mostra um exemplo de diagrama de contexto.

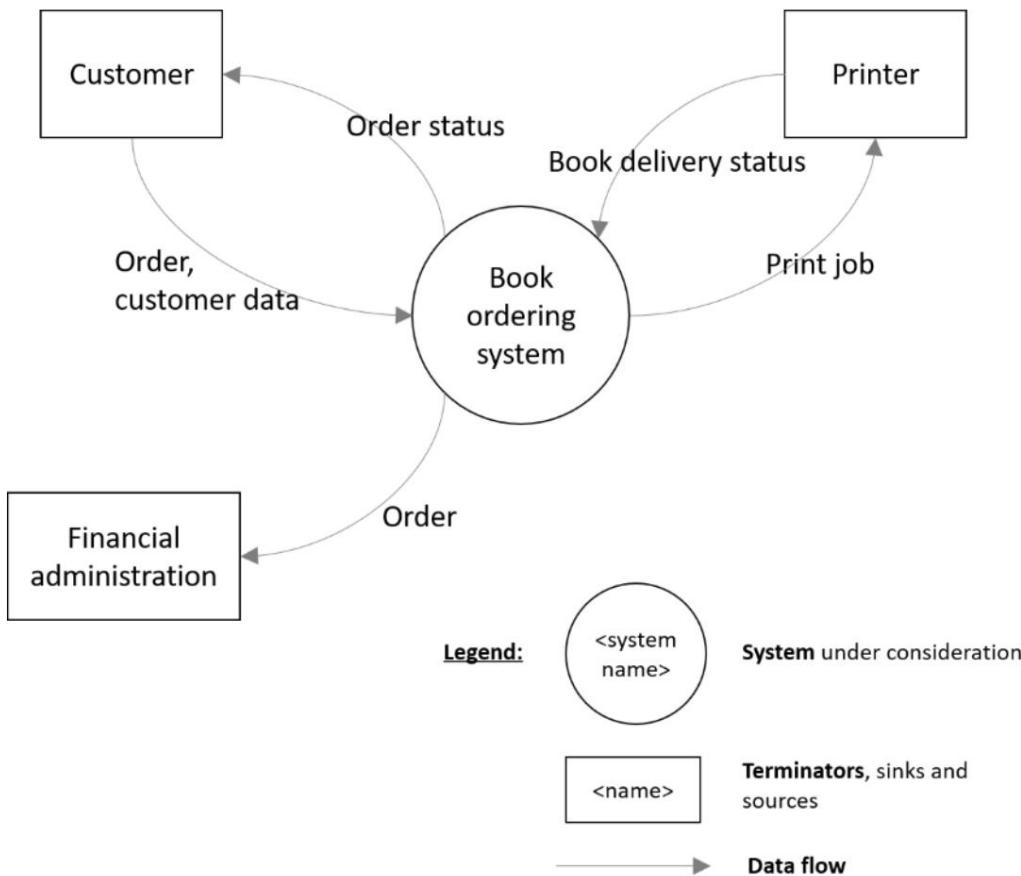


Figura 3.5 Exemplo de um diagrama de contexto usando um DFD

O sistema é colocado centralmente no modelo. Tem um nome claro para que os leitores saibam qual sistema está sendo considerado.

Os retângulos ao redor do sistema são terminadores: cliente, impressora e administração financeira. Um terminador que fornece informações ou serviços ao sistema é chamado de *fonte*. Um terminador que obtém informações ou serviços do sistema é chamado de *coletor*. Um terminador pode assumir qualquer função, dependendo dos dados fornecidos ou recuperados, como o cliente no exemplo acima.

As setas no exemplo mostram como as informações dos terminadores fluem para o sistema (origem) e do sistema para os terminadores (sumidouros). As setas recebem um nome lógico que descreve quais informações são transferidas. Detalhes irrelevantes são omitidos no nível do diagrama de contexto. O fluxo de informações entre o cliente e o sistema contém, por exemplo, *dados do cliente*. Quais informações (nome, data de nascimento, endereço de e-mail, número de telefone, endereço de entrega, endereço de cobrança etc.) compõem os *dados do cliente* não precisam ser relevantes ainda para esse nível de abstração.

Um DFD fornece informações sobre a interface do sistema com seu contexto.

O fluxo de informações pode consistir em objetos tangíveis (materiais) e intangíveis (informações). Além disso, nesse nível conceitual, não há referência (ainda) a como — e-mail, site, formulário etc. — as informações são fornecidas.

Adicionar detalhes extras ao diagrama de contexto pode torná-lo mais claro para as partes interessadas envolvidas e pode ajudar a melhorar o entendimento compartilhado. Esses detalhes precisam ser elaborados para cada situação individual.

Usar um diagrama de fluxo de dados para modelar o contexto de um sistema fornece alguns insights sobre as interações do sistema com seu ambiente, por exemplo:

- As interfaces com pessoas, departamentos, organizações e outros sistemas no ambiente
- Os objetos (tangíveis e intangíveis) que o sistema recebe do ambiente
- Os objetos (tangíveis e intangíveis) que são produzidos pelo sistema e são entregues ao ambiente

Um diagrama de fluxo de dados indica um limite claro entre o sistema e seu ambiente. Os usuários e sistemas relevantes do ambiente são identificados durante a elicitação de requisitos (Seção 4.1). Os diagramas de contexto DFD podem ajudar a estruturar o contexto para alcançar um entendimento compartilhado do contexto do sistema e do limite do sistema.

3.4.2.2 Diagrama de Caso de UML

Outra visão do contexto de um sistema pode ser alcançada a partir de uma perspectiva funcional. O diagrama de caso de uso UML é uma abordagem comum para modelar os aspectos funcionais de um sistema e os limites do sistema, juntamente com as interações do sistema com usuários e outros sistemas. Os casos de uso fornecem uma maneira fácil de descrever sistematicamente as várias funções dentro do escopo definido da perspectiva do usuário. Isso é diferente dos diagramas de contexto DFD, onde o sistema é representado como uma grande caixa preta.

Os casos de uso foram propostos pela primeira vez como um método para documentar as funções de um sistema em [Jaco1992]. Os casos de uso da UML consistem em diagramas de caso de uso com especificações de caso de uso textuais associadas (consulte a Seção 3.3.2). Uma especificação de caso de uso especifica cada caso de uso em detalhes, por exemplo, descrevendo as possíveis atividades do caso de uso, sua lógica de processamento e pré-condições e pós-condições da execução do caso de uso. A especificação de casos de uso é essencialmente textual - por exemplo, por meio de modelos de casos de uso, conforme recomendado em [Cock2001].

Conforme mencionado, um diagrama de caso de uso UML mostra as funções (casos de uso) do ponto de vista dos usuários diretos e outros sistemas que interagem com o sistema em questão. O nome do caso de uso geralmente é composto de um verbo e um substantivo. Isso dá uma breve descrição da função oferecida pelo sistema, conforme o exemplo da Figura 3.6.

Um diagrama de caso de uso fornece informações sobre a funcionalidade fornecida aos usuários diretos no contexto

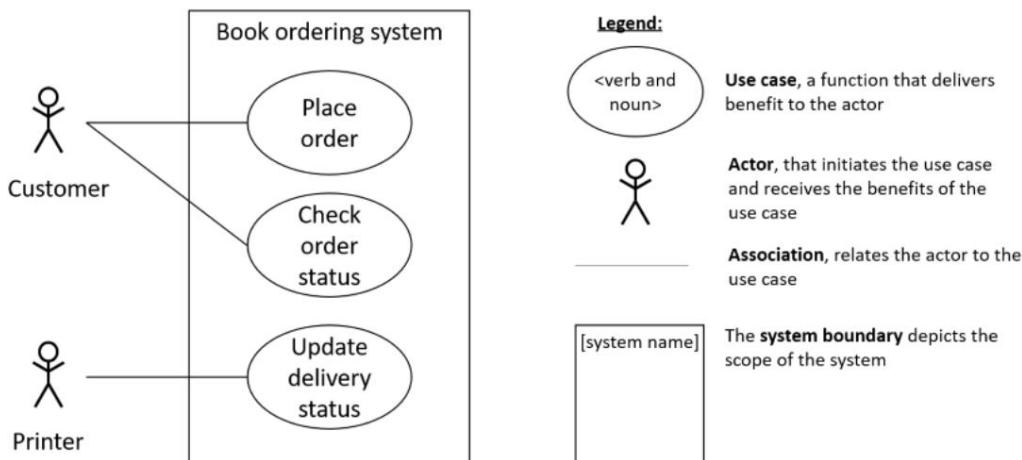


Figura 3.6 Exemplo de um diagrama de contexto usando um diagrama de caso de uso UML

Os atores são os usuários diretos ou sistemas que interagem com o sistema em consideração. O ator (usuário ou sistema) que inicia o caso de uso recebe o benefício que o caso de uso oferece (por exemplo, mostrar o status de um pedido ao cliente). A associação conecta o ator com o caso de uso relevante, mas não documenta nenhuma direção ou fluxo de dados (como é feito nos DFDs); expressa apenas que o ator recebe o benefício do caso de uso.

Um diagrama de caso de uso UML descreve a funcionalidade que o sistema oferece ao seu ambiente. A separação entre a funcionalidade no sistema e os atores no contexto é visualizada com o limite do sistema (retângulo ao redor dos casos de uso, por exemplo, "sistema de pedidos de livros"). Os diagramas de caso de uso suportam a nitidez do limite do sistema e verificam se o escopo funcional do sistema em um alto nível é coberto.

Cada caso de uso também inclui uma especificação detalhada do caso de uso, documentando as pré-condições, gatilhos, ações, pós-condições, atores e assim por diante. Os casos de uso geralmente são descritos usando um modelo (Seção 3.3). Se os cenários de um caso de uso se tornarem complexos ou grandes, a recomendação é visualizar os cenários com diagramas de atividades UML, consulte a Seção 3.4.4.1. A especificação detalhada dos casos de uso não faz parte da modelagem de contexto e pode ser elaborada posteriormente, quando esta informação se tornar relevante.

3.4.3 Estrutura de Modelagem e Dados

Para requisitos funcionais da perspectiva de objetos de negócios (consulte a Seção 3.1.4), diferentes modelos de dados estão disponíveis. Um objeto (negócio) pode ser um objeto tangível ou intangível, como uma bicicleta, pedal, campainha de bicicleta, mas também uma solicitação de treinamento, uma cesta de compras com produtos digitais e assim por diante. Um objeto (de negócios) é "algo" no mundo real. Alguns (ou talvez todos) desses objetos (de negócios) são usados pelo sistema em questão. O sistema usa esses objetos como entrada para processar, persistir e/ou entregar saída. Os modelos de dados são usados para descrever os objetos (de negócios) que devem ser conhecidos pelo sistema. Esses tipos de diagramas modelam o objeto, os atributos do objeto e os relacionamentos entre os objetos. Por uma questão de simplicidade, nos referimos à estrutura e dados de modelagem - estes, no entanto, representam estruturas de informações entre objetos (de negócios) no mundo real.

Uma série de modelos comuns para representar a estrutura e os dados são:

- Diagramas de relacionamento de entidade (ERD) [Chen1976]
- Diagramas de classe UML [OMG2017]. Consulte a Seção 3.4.3.1
- Diagramas de definição de bloco SysML [OMG2018]. Consulte a Seção 3.4.6.2

Modelos comuns para representar estrutura e dados

Para explicar o conceito de modelagem de estrutura e dados, este capítulo usa o diagrama de classe UML como exemplo. UML, abreviação de Unified Modeling Language, consiste em um conjunto integrado de diagramas. Este conjunto de diagramas é uma coleção das melhores práticas de engenharia e provou ser bem-sucedido na modelagem de sistemas grandes e complexos. A UML foi projetada por Grady Booch, James Rumbaugh e Ivar Jacobson na década de 1990 e tem sido uma linguagem de modelagem padronizada desde 1997. Se mais profundidade ou um modelo diferente for desejado, leia a literatura mencionada e pratique com a linguagem de modelagem desejada.

3.4.3.1 Diagramas de Classe UML

UML é uma coleção de diferentes modelos que podem ser usados para descrever um sistema. Um desses modelos é o diagrama de classes. Um diagrama de classes descreve um conjunto de classes e associações entre elas. Discutimos apenas a notação comum e simples

elementos deste modelo. Se desejar mais profundidade, consulte a literatura ou a Modelagem de requisitos de nível avançado do CPRE.

Na visão geral abaixo, você encontrará os elementos de notação mais comuns.

Elementos de notação mais comuns de diagramas de classe UML

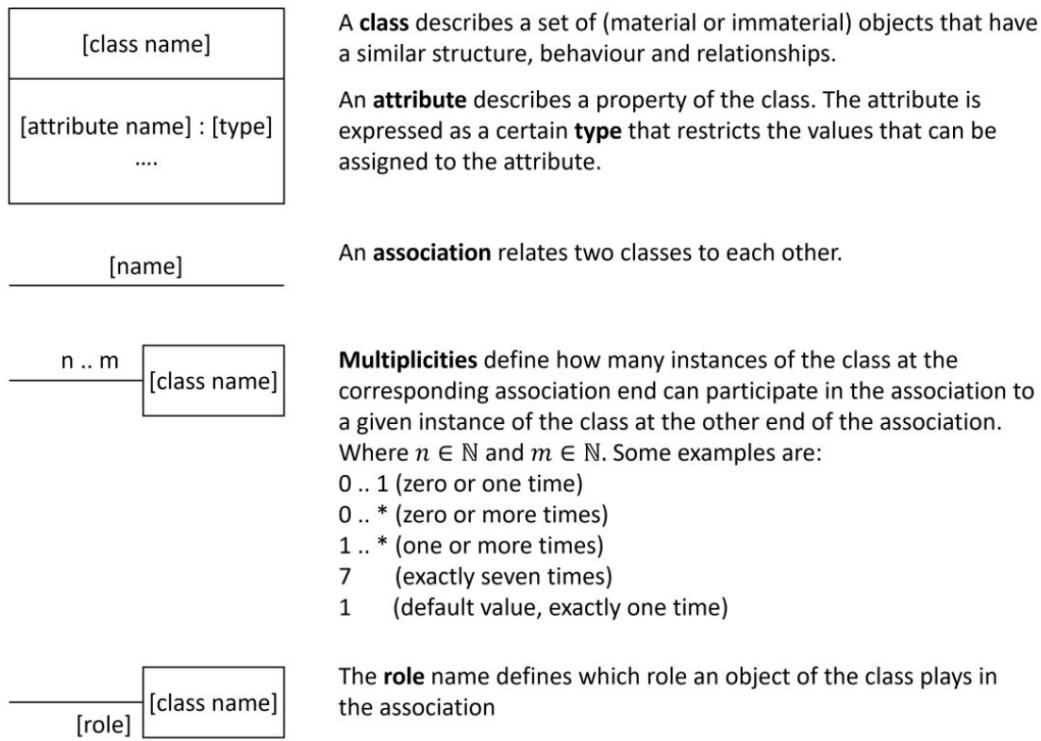


Figura 3.7 Subconjunto dos elementos de modelagem dos diagramas de classe UML

Em um modelo de classe, você encontrará os conceitos e termos relevantes no domínio. Esses conceitos incluem uma definição clara incluída no glossário. Com o uso de modelos de dados, o glossário é ampliado com informações sobre a estrutura e coerência dos termos e conceitos. Uma definição clara e coerente dos termos usados evita a falta de comunicação sobre o assunto em consideração.

A Figura 3.8 mostra um modelo simplificado do sistema de pedidos de livros (veja exemplos do contexto na Seção 3.4.2). As informações estáticas de que o sistema precisa para executar sua funcionalidade – solicitar um livro – são modeladas.

Um cliente pede um livro e, portanto, as informações são mantidas para as classes *Cliente*, *Pedido* e *Livro*. Um cliente pode fazer um pedido e, portanto, existe um relacionamento (associação) entre o *cliente* e o *pedido*. Um cliente pode fazer vários pedidos ao longo do tempo e só se torna cliente se fizer um pedido. Esta informação determina a multiplicidade: 1 cliente faz 1 ou mais pedidos.

O fato de um cliente poder solicitar um livro significa que também existe um relacionamento entre as classes *Pedido* e *Livro*. Para manter o exemplo simples, aqui o cliente pode solicitar apenas um livro por vez. Além disso, um pedido deve conter pelo menos um livro. Uma ordem que não tem livro não é uma ordem.

Na classe *Book*, o atributo *inStock* também é mantido. Informações como "se o estoque não for suficiente para atender ao pedido, um trabalho de impressão será enviado para a impressora" não podem ser modeladas. Este é um tipo de informação que não pode ser modelado em um diagrama de classes porque descreve uma determinada funcionalidade do sistema. Essas informações fazem parte dos requisitos e devem ser documentadas em outro produto de trabalho. Pode ser

adicionado como um requisito textual que acompanha o diagrama de classes ou pode ser modelado com outro diagrama - por exemplo, um diagrama de atividades UML (consulte a Seção 3.4.4.1).

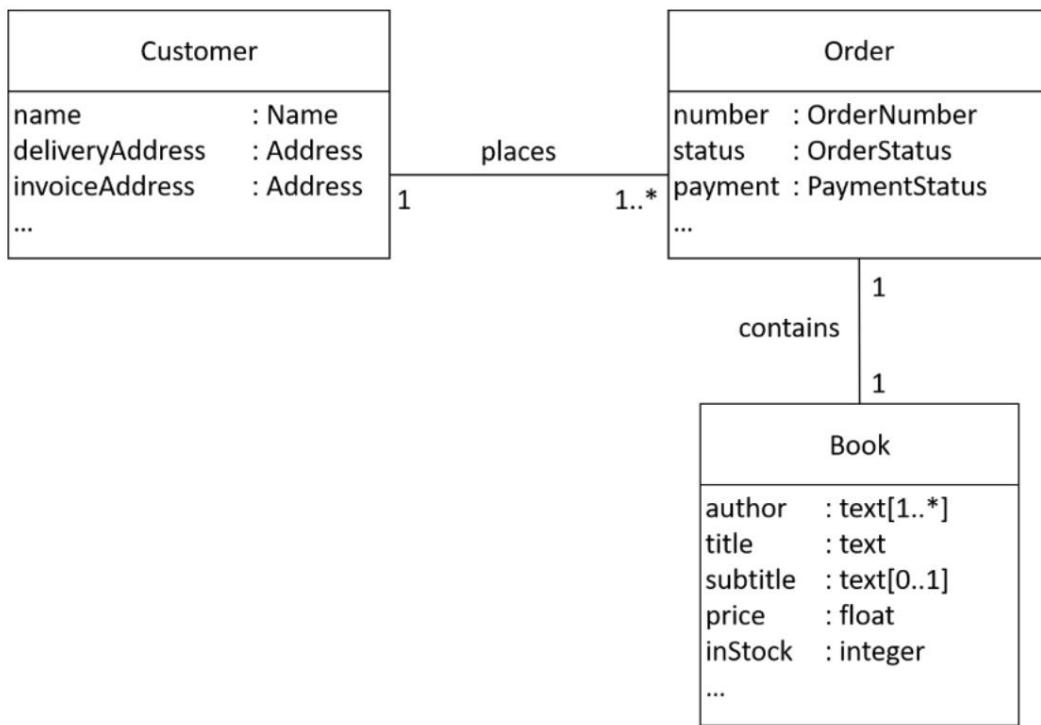


Figura 3.8 Exemplo de um diagrama de classe UML simples

3.4.4 Função e Fluxo de Modelagem

Função e fluxo descrevem como o (sub)sistema deve transformar entrada em saída.

Podemos visualizar esse tipo de requisito com modelos que retratam a função e o fluxo.

Ao contrário dos dados de modelagem, que precisam essencialmente de apenas um tipo de diagrama, a função e o fluxo podem ser vistos de diferentes ângulos. Dependendo das necessidades das partes interessadas para dar o próximo passo no processo de desenvolvimento, mais de um modelo pode ser necessário para documentar os requisitos sobre função e fluxo.

Alguns modelos comuns para representar a função e o fluxo são:

- Diagrama de caso de uso UML [OMG2017]. Consulte a Seção 3.4.2.2
- Diagrama de atividades UML [OMG2017]. Consulte a Seção 3.4.4.1
- Diagrama de fluxo de dados [DeMa1978]. Consulte a Seção 3.4.2.1
- Modelos de história de domínio [HoSch2020]. Consulte a Seção 3.4.6.3
- Notação de modelagem de processos de negócios (BPMN) [OMG2013].

Modelos comuns para representar a função e o fluxo

Excuse: Modelos de processo BPMN são usados para descrever processos de negócios ou processos técnicos. BPMN é freqüentemente usado para expressar modelos de processos de negócios.

Para explicar o conceito de função e fluxo de modelagem, limitamos esta seção a alguns exemplos de diagramas UML. Se desejar mais profundidade ou um modelo diferente, leia a literatura mencionada e pratique com a linguagem de modelagem relevante.

3.4.4.1 Diagrama de Atividades UML

Os modelos de atividade UML são usados para especificar as funções do sistema. Eles fornecem elementos para modelar ações e controlar o fluxo entre as ações. Os diagramas de atividades também podem expressar quem é responsável por qual ação. Elementos de modelagem avançados (não cobertos por este manual) fornecem os meios para modelar o fluxo de dados.

Um diagrama de atividades UML expressa o fluxo de controle das atividades de um (sub)sistema. O pensamento de fluxo vem da visualização do código do programa com fluxogramas (de acordo com [DIN66001], [ISO5807]). Isso ajudou os programadores a conceber e entender estruturas e fluxos complexos em programas. Com a introdução da UML [OMG2017], foi introduzido um modelo para visualizar atividades e ações de uma perspectiva funcional.

Na visão geral abaixo, você encontrará os elementos básicos de notação.

Elementos básicos de notação de diagramas de atividades UML



The **start node** indicates the starting point of the activity diagram.



The **end node** indicates the end point of the activity flow. There can be multiple end nodes.



The **control flow termination** ends the activity flow.



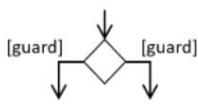
An **action/activity** represents the non-interruptible action of objects. An executable computation that results in a change in state of the system or the return of a result.



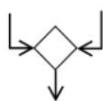
The **action or control flow** illustrates the transitions from one action state to another (also called edges and paths).

Figura 3.9 Elementos de notação básica do diagrama de atividades UML

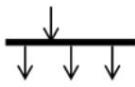
Com este conjunto de elementos básicos de notação, você pode configurar um diagrama de atividade sequencial simples. Se for necessário mais controle, o modelo pode ser estendido com decisões e fluxos paralelos de atividades usando os elementos de notação abaixo.



The **decision node** branches the activity flow based on a guard. The outgoing alternatives should be labeled with a condition or guard expression. The guard must be true before the move to the appropriate flow.

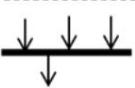


The **merge node** brings together multiple flows that are not concurrent.



Synchronization bar

A **fork node** is used to split a single incoming flow into multiple concurrent flows.



A **join node** joins multiple concurrent flows back into a single outgoing flow. The action following the join node is executed after all parallel actions have been executed.

A fork and join mode are used together and are therefore often referred to as synchronization.

Figura 3.10 Decisões e fluxos paralelos em um diagrama de atividades UML

Os diagramas de atividades podem ser usados para especificar a lógica de processamento dos cenários de caso de uso em detalhes (consulte a Seção 3.3.2). Os diagramas de atividades são criados para visualizar os cenários, que são processos com atividades e lógica de processamento. Desde que o diagrama permaneça compreensível, o cenário principal pode ser modelado em conjunto com os cenários alternativos e os cenários de exceção como parte do mesmo diagrama.

A Figura 3.11 dá um exemplo simples do sistema de pedidos de livros. Esse fluxo simplificado de ação começa quando o cliente envia seu pedido. Primeiro, as informações do *Pedido* e do *Cliente* são validadas para determinar se todas as informações (necessárias) foram fornecidas. Se as informações do *Pedido* ou do *Cliente* forem inválidas (incorrectas ou insuficientes), uma notificação será enviada ao cliente e o processo do pedido será cancelado. O cenário básico é que as informações do *Pedido* e do *Cliente* são válidas. O cenário em que as informações do *Pedido* ou do *Cliente* são inválidas é chamado de fluxo excepcional e lida com uma condição de falha funcional no processo.

Se as informações do *Pedido* e do *Cliente* estiverem corretas, o estoque é verificado. Havendo quantidade suficiente de produtos em stock, a *Encomenda* é recolhida e enviada ao *Cliente*. Um fluxo alternativo é iniciado se houver produtos insuficientes em estoque. Uma solicitação de trabalho de impressão é enviada para a *impressora* e uma notificação para uma nova entrega é enviada para o *Cliente*.

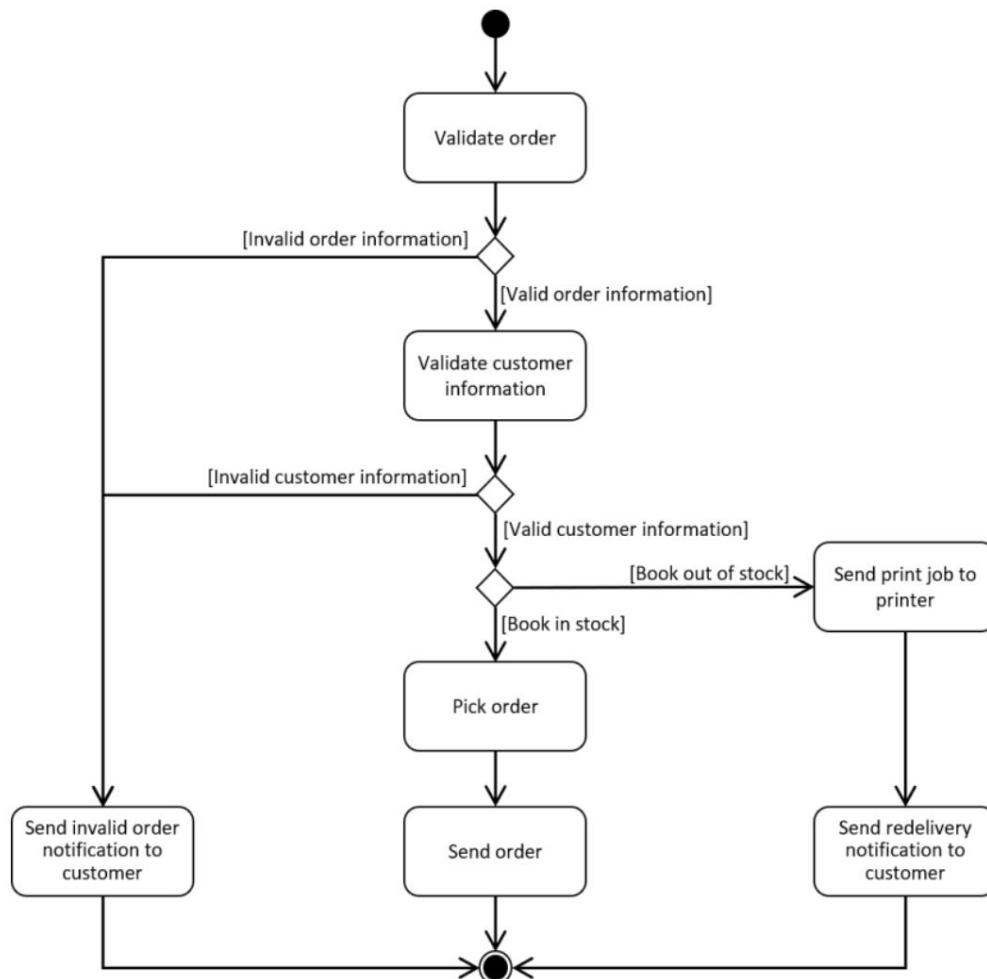


Figura 3.11 Exemplo de um diagrama de atividades UML

Dentro do sistema de pedidos de livros, também existem outros fluxos separados do processo de pedido e entrega. Por exemplo, os processos de pagamento, devolução e fatura têm fluxos separados para permitir uma clara separação de preocupações. Se, por exemplo,

a decisão é tomada para não manter nenhum produto em estoque, então o processo de pedido e entrega ainda se aplica. Se forem necessárias alterações neste fluxo, essas alterações podem não afetar os outros fluxos. Essa decomposição de funcionalidade ajuda a manter as coisas simples e claras.

3.4.5 Modelando Estado e Comportamento

Requisitos funcionais que descrevem o comportamento, estados e transições de um (sub)sistema ou de um objeto de negócios são requisitos na perspectiva comportamental. Um exemplo de estado do sistema é *On*, *Standby* ou *Off*. Um objeto de negócios pode ter um ciclo de vida que passa por vários estados prescritos. Por exemplo, um *Pedido* de objeto de negócios pode estar nos seguintes estados: *Colocado*, *Validado*, *Pago*, *Enviado* e *Concluído*.

Uma técnica amplamente utilizada para descrever o comportamento de um sistema são os gráficos de estado [Hare1988]. Statecharts são máquinas de estado com estados que são decompostos hierarquicamente e/ou ortogonalmente. Máquinas de estado, incluindo gráficos de estado, podem ser expressas na linguagem de modelagem UML [OMG2017] com diagramas de máquina de estado (também chamados de diagramas de estado).

Diagramas de estado descrevem máquinas de estado que são finitas. Isso significa que esses sistemas eventualmente atingem um estado final. Um diagrama de estado mostra os estados que o sistema ou um objeto pode assumir. Também indica como mudar de estado, ou seja, a transição de estado. Um sistema faz pouco por si só. A mudança de estado requer um acionador do sistema ou do ambiente do sistema.

Modelos comuns para representar comportamento e estados incluem:

- ▶ Gráficos de estado [Hare1988]
- ▶ Diagrama de estado UML [OMG2017]

3.4.5.1 Diagrama de estado UML

Para explicar o conceito de comportamento e estados de modelagem, este capítulo usa o diagrama de estado UML como exemplo. Se desejar mais profundidade ou um modelo diferente, leia a literatura mencionada e pratique com a linguagem de modelagem relevante.

Na visão geral abaixo, você encontrará os elementos básicos de notação.

Elementos básicos de notação de diagramas de estado UML

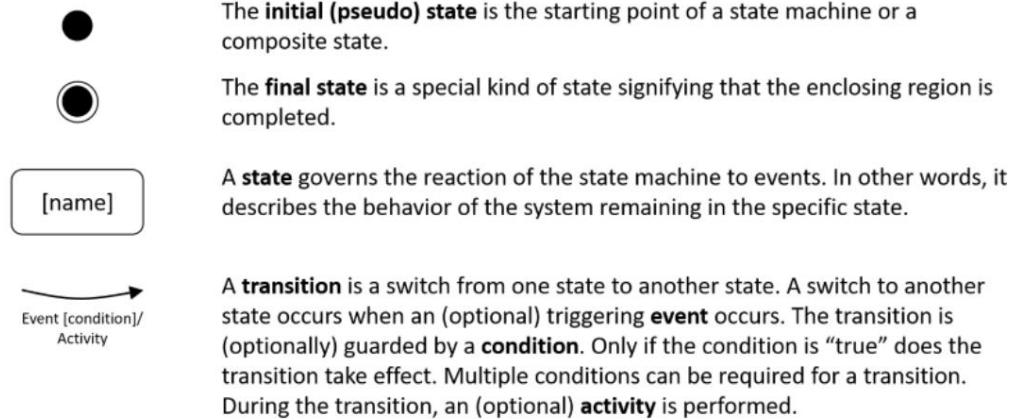


Figura 3.12 Elementos de notação básica do diagrama de estado UML

Conforme discutido no início da seção, um diagrama de estado pode esclarecer os estados que um objeto pode assumir. Vemos aqui uma oportunidade de visualizar informações adicionais (e parcialmente redundantes) de um objeto. Imagine que você encomendou um livro em um site e deseja acompanhar o status do seu pedido. Um pedido é usado no mundo real e é modelado como um objeto de negócios em um diagrama de classes (consulte a Figura 3.8) com, provavelmente, um *status de atributo*. O diagrama de classes indica que o *status* do atributo pode assumir um número limitado de valores, como *Validado*, *Pago*, *Entregue*, *Cancelado* e assim por diante. O diagrama de classes não descreve a ordem das possíveis mudanças de status. Um diagrama de classes também não descreve o comportamento do sistema em um determinado "status". Isso pode ficar claro com um diagrama de estado UML - por exemplo, que um pedido oferecido não pode ir diretamente para o status *Entregue* sem que o cliente tenha pago pelo pedido.

A Figura 3.13 dá um exemplo de diagrama de estado do sistema de pedidos de livros. No diagrama de classes (Figura 3.8) do sistema de pedidos de livros, um objeto *Order* é modelado.

Este objeto possui um *status* de atributo que pode ter um número limitado de valores. Esses valores são listados e explicados no diagrama de classes. O que um diagrama de classes não descreve é a sequência na qual o pedido é processado. Um diagrama de estado visualiza os estados e as transições entre os estados, deixando claro qual é a sequência do status do pedido. O diagrama de estado mostra, por exemplo, que o pedido não pode ser enviado antes de ser completamente coletado (transição entre os estados *Picked* e *Enviado*).

Além disso, se o pedido estiver no estado *Enviado*, o próximo estado só poderá ser *Pago*. Uma transição de *Enviado* para *Manuseado* não é possível. Este diagrama também deixa claro que o pagamento acontece após o envio do livro. Você pode perguntar às partes interessadas se é isso que elas precisam ou solicitaram.

Uma transição pode direcionar para o mesmo status. Esta situação é visível no estado *Picked*. Cada vez que o pedido não é coletado até a conclusão, ele permanece no mesmo estado para evitar que envie um pedido incompleto. Somente quando o pedido é totalmente separado, ele é enviado ao cliente.

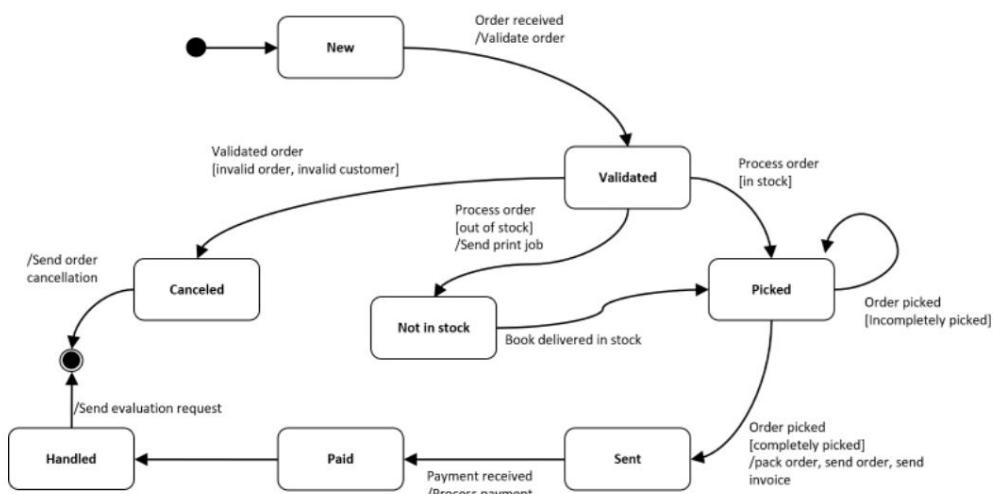


Figura 3.13 Exemplo de um diagrama de estado UML

Alguns meses após o lançamento do sistema de pedidos de livros, os clientes reclamaram que não podiam cancelar um pedido. Foi acordado que um cliente poderia cancelar o pedido em cada estado do processo de pedido. A modelagem desse novo requisito significa que uma transição para *Cancelado* é necessária em cada estado. Isso pode dificultar a leitura do diagrama. Adicionar um requisito textual para descrever esse comportamento pode ser uma forma de manter o modelo simples para o público.

3.4.6 Modelos complementares

No nível básico do CPRE, o entendimento e a aplicação dos modelos são restritos a tipos de modelos importantes e selecionados. Existem outros tipos de modelo que são usados na Engenharia de Requisitos. As subseções a seguir fornecem alguns exemplos adicionais para modelos que são complementares e não serão questionados no exame de nível CPRE Foundation.

3.4.6.1 Metas de Modelagem

Os requisitos de negócios descrevem uma meta ou necessidade de negócios. Eles descrevem o resultado final que a solução deve atingir e com o qual o problema (de negócios) é resolvido, consulte o Capítulo 2, Princípio 5. Para garantir que o foco esteja na solução do problema e que o esforço se concentre em agregar valor, as metas são cuidadosamente descrito. Na Engenharia de Requisitos, existem várias maneiras de documentar metas. O mais comum é o uso de linguagem natural (Seção 3.2) ou modelos (Seção 3.3). Formulários de documentação baseados em modelos podem ser encontrados, por exemplo, em [Pich2010], [Pohl2010] ou [RoRo2012].

Existem também algumas notações baseadas em modelos para documentar objetivos. A notação mais fácil e comum é uma árvore AND/OR [AnPC1994]. As árvores AND/OR nos permitem documentar metas em diferentes níveis de detalhe e vincular submetas a metas usando relacionamentos AND e OR. Uma relação AND significa que todas as submetas precisam ser cumpridas para cumprir a meta. Uma relação OU é usada para expressar que pelo menos uma das submetas precisa ser cumprida para cumprir a meta.

Abordagens de modelagem mais elaboradas para metas podem ser encontradas em:

- Linguagem de requisitos orientados a objetivos (GRL) [GRL2020]

Esta é uma linguagem que suporta modelagem orientada a objetivos e raciocínio de requisitos, especialmente para lidar com requisitos não funcionais.
- Aquisição de conhecimento em especificação automatizada (KAOS) [vLam2009]

KAOS é uma metodologia que contém modelagem de objetivos. Isso permite que os analistas criem modelos de requisitos e obtenham documentos de requisitos dos modelos de objetivos da KAOS.
- A estrutura i* é um dos métodos de raciocínio e modelagem orientados a objetivos e agentes mais populares no campo. i* suporta a criação de modelos que representam uma organização ou um sistema sócio-técnico.

[FLCC2016] fornece uma visão geral abrangente da estrutura i* e sua aplicação.

A documentação dos objetivos (em forma textual ou gráfica) é um importante ponto de partida para eliciar requisitos, referindo-se aos requisitos à sua lógica e identificando fontes – como partes interessadas – dos requisitos, etc.

3.4.6.2 Diagramas de definição de blocos SysML

A linguagem de modelagem de sistemas (SysML) [OMG2018] é uma linguagem de modelagem de uso geral para aplicativos de engenharia de sistemas. SysML é um dialeto da UML, que reutiliza e estende partes da UML.

SysML pode ser adotado para muitos propósitos diferentes. *Diagramas de definição de bloco* em SysML são uma extensão do diagrama de classe UML. Eles podem, por exemplo, ser adaptados para expressar diagramas de contexto usando blocos estereotipados para o sistema e os atores. Diagramas de definição de bloco também podem ser usados para modelar a estrutura de um sistema em termos de entidades conceituais do sistema e as relações entre elas.

3.4.6.3 Modelos de história de domínio

Modelos de história de domínio podem ser usados para modelar função e fluxo, especificando histórias visuais sobre como os atores interagem com dispositivos, artefatos e outros itens em um domínio, normalmente usando símbolos específicos de domínio [HoSch2020]. Eles são um meio para entender o domínio de aplicação no qual um sistema irá operar.

As técnicas devem ser executáveis de forma muito simples para contar uma história, um quadro e notas adesivas podem ser suficientes. Reunir as partes interessadas relevantes que realmente sabem como o negócio opera provoca discussões significativas ao contar histórias que ocorrem no domínio. A narrativa de domínio melhora a compreensão compartilhada de um processo de negócios e é usada para analisar e resolver problemas no domínio.

3.4.6.4 Diagrama de Sequência UML

O diagrama de seqüência UML é usado para representar a interação entre os parceiros de comunicação e para modelar o aspecto dinâmico dos sistemas. O aspecto dinâmico dos sistemas que um diagrama de seqüência representa pode ser função e fluxo, bem como estado e comportamento. Portanto, um diagrama de sequência UML pode ser usado para diferentes propósitos.

Os parceiros de comunicação em um diagrama de seqüência UML são atores, sistemas, componentes e/ou objetos dentro de um sistema. A interação exibe a seqüência de mensagens (um cenário) entre esses parceiros de comunicação. A interação que ocorre entre os parceiros de comunicação realiza o propósito de um cenário, respectivamente (uma parte) de um caso de uso.

Na visão geral abaixo, você encontrará os elementos básicos de notação.

Elementos básicos de notação de diagramas de sequência UML

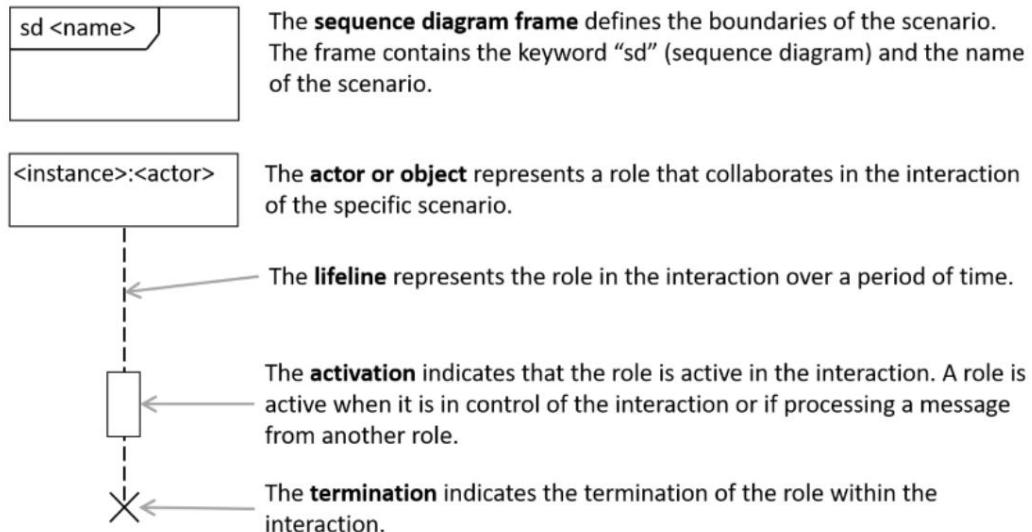


Figura 3.14 Elementos de notação básica do diagrama de seqüência UML

Uma linha de vida em um cenário descreve o papel no cenário, ou seja, a instância de um ator. Quando os diagramas de sequência são modelados, o nome da instância de um ator ou objeto geralmente é omitido. Os papéis que participam da comunicação interagem entre si enviando mensagens. Existem dois tipos de mensagens que são usadas na interação.

Elementos básicos de notação de mensagens em diagramas de sequência UML

- The **asynchronous message** represents a message without the sender having to wait for an answer.
- The **synchronous message** represents a message for which the sender waits for an answer.
- ↔ The **answer message** for a synchronous message.

Figura 3.15 Elementos de notação básica de mensagens no diagrama de sequência UML

Uma mensagem também pode ser enviada de ou para objetos fora do cenário. Isso é representado como um círculo preenchido. O remetente ou destinatário desses tipos de mensagens pode ser desconhecido.

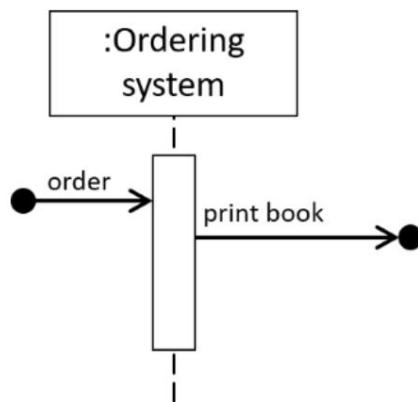


Figura 3.16 Mensagens de e para um objeto fora do cenário

A Figura 3.17 mostra um modelo do cenário em que um *cliente* pede um livro e esse livro específico está esgotado. O *Cliente* pede para fazer um *Pedido*. Se o *Pedido* for inválido, uma notificação informando que o *Pedido* foi cancelado é retornada. Se a *Encomenda* for válida, verifica-se o stock e se faltar um livro, é enviado um trabalho de impressão para a *Impressora*. Esta é uma mensagem síncrona porque estamos aguardando o recebimento do livro – mesmo que demore um pouco para imprimir o livro. Uma notificação é enviada ao *Cliente* informando que o livro está esgotado e será reenviado. A *Encomenda* fica desativada até que o livro seja entregue pela *Gráfica*.

Quando o livro é recebido da *Impressora*, o *sistema de Pedidos* é ativado novamente. A encomenda é recolhida e enviada para o *Cliente*. Isso conclui o *Pedido* e uma última notificação do status é enviada ao *Cliente*.

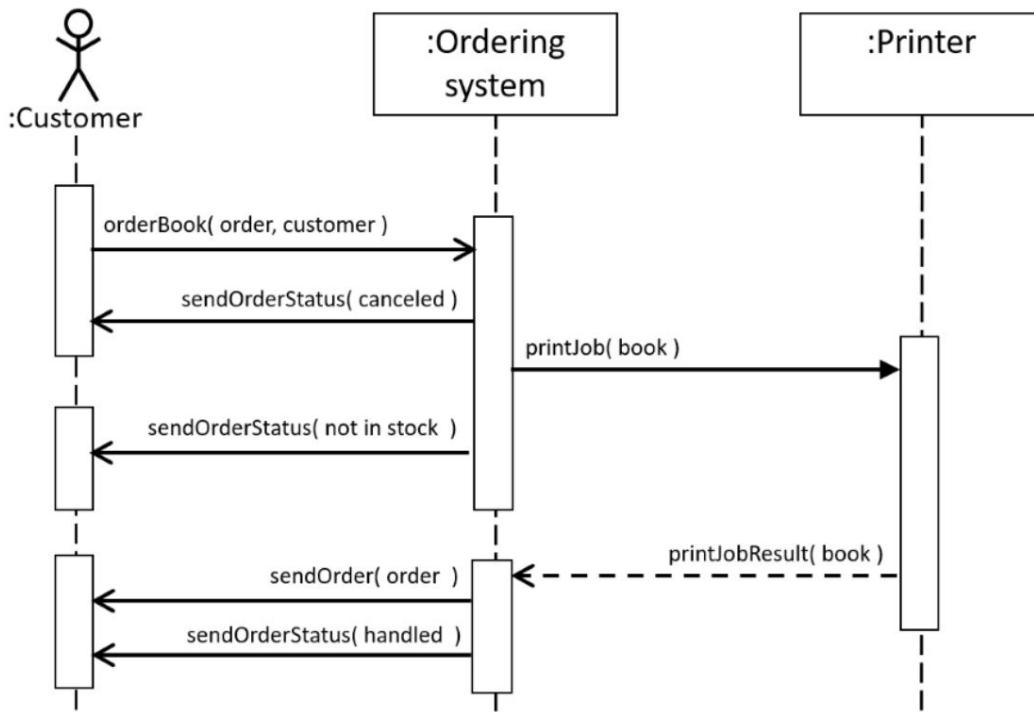


Figura 3.17 Exemplo de um diagrama de sequência UML

3.5 Glossários

Os glossários são um meio essencial de estabelecer um entendimento compartilhado da terminologia usada no desenvolvimento de um sistema: eles ajudam a evitar que as pessoas envolvidas como partes interessadas ou desenvolvedores usem e interpretem os mesmos termos de maneiras diferentes.

[Por que glossários](#)

Um bom glossário contém definições para todos os termos relevantes para o sistema, sejam termos específicos do contexto ou termos cotidianos que são usados com um significado especial no contexto do sistema a ser desenvolvido. Um glossário também deve definir todas as abreviaturas e acrônimos usados. Se houver sinônimos (ou seja, termos diferentes que denotam a mesma coisa), eles devem ser marcados como tal. Homônimos (isto é, termos idênticos que denotam coisas diferentes) devem ser evitados ou pelo menos marcados como tal no glossário.

[conteúdo do glossário](#)

Existem algumas regras que orientam a criação, uso e manutenção do glossário em um projeto de desenvolvimento de sistema.

[regras do glossário](#)

- **Criação e manutenção.** Para garantir que a terminologia definida no glossário seja consistente e esteja sempre atualizada, é vital que o glossário seja gerenciado e mantido centralmente ao longo de todo o projeto, com uma pessoa ou um pequeno grupo sendo responsável pelo glossário. Ao definir os termos, é importante que as partes interessadas estejam envolvidas e concordem com a terminologia.

- **Uso.** Para obter todos os benefícios de um glossário, seu uso deve ser obrigatório. Os produtos de trabalho devem ser verificados quanto ao uso adequado do glossário. Obviamente, isso significa que todos os envolvidos em um projeto devem ter acesso de leitura ao glossário.

Quando uma organização desenvolve sistemas relacionados em vários projetos, faz sentido criar um glossário no nível da empresa para obter uma terminologia consistente entre os projetos.

Criar, manter e usar um glossário de forma consistente evita erros e mal-entendidos sobre a terminologia utilizada. Trabalhar com glossários é uma prática recomendada padrão em RE.

3.6 Documentos de Requisitos e Estruturas de Documentação

Não é suficiente trabalhar com requisitos no nível de requisitos individuais.

Os requisitos devem ser reunidos e agrupados em produtos de trabalho adequados, sejam eles documentos de requisitos explícitos ou outras estruturas de documentação relacionadas a ER (como um backlog de produto).

Modelos de documentos (consulte a Seção 3.3.3) podem ser usados para organizar esses documentos com uma estrutura bem definida, a fim de criar uma estrutura consistente e sustentável coleção de requisitos. Modelos de documentos estão disponíveis na literatura [Vole2020], [RoRo2012] e nos padrões [ISO29148]. Os modelos também podem ser reutilizados de projetos semelhantes anteriores ou podem ser impostos por um cliente. Uma organização também pode decidir criar um modelo de documento como um padrão interno.

Um documento de requisitos também pode conter informações e explicações adicionais - por exemplo, um glossário, critérios de aceitação, informações do projeto ou características da implementação técnica.

Os documentos de requisitos usados com frequência são:

- *Especificação dos requisitos das partes interessadas*: os desejos e necessidades das partes interessadas que devem ser satisfeitos pela construção de um sistema, visto da perspectiva das partes interessadas. Quando um cliente escreve uma especificação de requisitos das partes interessadas, ela é chamada de *especificação de requisitos do cliente*.
- *Especificação de requisitos do usuário*: um subconjunto de uma especificação de requisitos das partes interessadas, cobrindo apenas os requisitos das partes interessadas que são possíveis usuários de um sistema.
- *Especificação de requisitos do sistema*: os requisitos para um sistema a ser construído e seu contexto para que satisfaça os desejos e necessidades de seus stakeholders.
- *Especificação de requisitos de negócios*: as metas, objetivos e necessidades de negócios de uma organização que devem ser alcançados empregando um sistema (ou uma coleção de sistemas).
- *Documento de visão*: uma imaginação conceitual de um sistema futuro, descrevendo suas principais características e como ele criará valor para seus usuários.

As estruturas de documentação alternativas usadas com frequência são:

- *Backlog do produto*: uma lista priorizada de itens de trabalho, cobrindo todos os requisitos necessários e conhecidos para o produto
- *Sprint backlog*: um subconjunto selecionado de um product backlog com itens de trabalho que serão realizados na próxima iteração
- *Mapa de histórias*: uma organização visual bidimensional de histórias de usuários em um backlog de produto em relação ao tempo e ao conteúdo

Não há documento de requisitos padrão ou universal ou estrutura de documentação. Assim, documentos ou estruturas de documentação não devem ser reutilizados de projetos anteriores sem reflexão.

Documentos e <documentação estruturas

Modelos de documento

Documentos usados com frequência

Documentação usada com frequência estruturas

Escolhendo um formulário de documentação adequado

A escolha real depende de vários fatores, por exemplo:

- O processo de desenvolvimento escolhido
- O tipo e o domínio do projeto (por exemplo, solução sob medida, desenvolvimento de produto ou personalização de produto padrão)
- O contrato (um cliente pode prescrever o uso de uma determinada estrutura de documentação)
- O tamanho do documento (quanto maior o documento, mais estrutura é necessária)

3.7 Protótipos na Engenharia de Requisitos

Os protótipos desempenham um papel importante tanto na engenharia quanto no design.

Definição 3.5 Protótipo: 1. Na fabricação: Uma peça construída antes do início da produção em massa. 2. Em engenharia de software e sistemas: Uma realização preliminar e parcial de certas características de um sistema. 3. No projeto: Uma instância preliminar e parcial de uma solução de projeto.

Protótipo

Protótipos em engenharia de software e sistemas são usados para três propósitos principais [LiSZ1994]:

Protótipos exploratórios são usados para criar entendimento compartilhado, esclarecer requisitos ou validar requisitos em diferentes níveis de fidelidade. Tais protótipos constituem produtos de trabalho temporários que são descartados após o uso. Protótipos exploratórios também podem ser usados como meio de especificação por exemplo. Esses protótipos devem ser tratados como produtos de trabalho em evolução ou duráveis.

Protótipo exploratório

Protótipos experimentais (também chamados de breadboards) são usados para explorar conceitos de soluções de design técnico, em particular no que diz respeito à sua viabilidade técnica. São descartados após o uso. Protótipos experimentais não são usados em ER.

Protótipo experimental

Protótipos evolutivos são sistemas piloto que formam o núcleo de um sistema a ser desenvolvido. O sistema final evolui estendendo e melhorando incrementalmente o sistema piloto em várias iterações. O desenvolvimento de sistemas ágeis freqüentemente emprega uma abordagem de prototipagem evolutiva.

protótipo evolutivo

Os Engenheiros de Requisitos usam principalmente protótipos exploratórios como meio de elicitação e validação de requisitos. Na elicitação, os protótipos servem como um meio de especificação por exemplo. Em particular, quando as partes interessadas não conseguem expressar claramente o que desejam, um protótipo pode demonstrar o que obteriam, o que os ajuda a moldar seus requisitos. Na validação, os protótipos são um meio poderoso para validar a *adequação* (ver Seção 3.8) dos requisitos.

Protótipos em ER

Protótipos exploratórios podem ser construídos e usados com diferentes graus de fidelidade. Distinguimos entre wireframes, mock-ups e protótipos nativos.

Estrutura de arame

Wireframes (também chamados de protótipos de papel) são protótipos de baixa fidelidade construídos com papel ou outros materiais simples que servem principalmente para discutir e validar ideias de design e conceitos de interface do usuário. Ao criar protótipos de sistemas digitais, os wireframes também podem ser construídos com ferramentas de esboço digital ou ferramentas de wireframing dedicadas. No entanto, ao usar uma ferramenta digital para wireframing, é importante manter as propriedades essenciais de um wireframe: ele pode ser construído rapidamente, modificado facilmente e não parece polido nem se assemelha a um produto final.

Mock-ups são protótipos de média fidelidade. Ao especificar sistemas digitais, eles usam telas reais e fluxos de cliques, mas sem funcionalidade real. Eles servem principalmente para especificar e validar interfaces de usuário. Os modelos oferecem aos usuários uma experiência realista de como interagir com um sistema por meio de sua interface de usuário. Eles são normalmente construídos com ferramentas de prototipagem dedicadas.

[Brincar](#)

Protótipos nativos são protótipos de alta fidelidade que implementam partes críticas de um sistema de forma que as partes interessadas possam usar o protótipo para ver se a parte prototipada do sistema funcionará e se comportará conforme o esperado.

[Protótipo nativo](#)

Eles servem tanto para especificação quanto para validação completa de requisitos críticos. Protótipos nativos também podem ser usados para explorar e decidir sobre *variantes* de requisitos para algum aspecto - por exemplo, duas formas diferentes possíveis de suportar um determinado processo de negócios.

Dependendo do grau de fidelidade, os protótipos exploratórios podem ser um produto de trabalho caro. Os engenheiros de requisitos devem considerar o compromisso entre o custo de construção e uso de protótipos e o valor obtido em termos de elicitação mais fácil e risco reduzido de requisitos inadequados ou mesmo errados.

3.8 Critérios de Qualidade para Produtos de Trabalho e Requisitos

Obviamente, os Engenheiros de Requisitos devem se esforçar para escrever bons requisitos que atendam a determinados critérios de qualidade. A literatura e os padrões de ER fornecem um rico conjunto de tais critérios de qualidade. No entanto, não há um consenso geral sobre quais critérios de qualidade devem ser aplicados aos requisitos. O conjunto de critérios apresentados nesta subseção visa fornecer uma prática comprovada em nível de fundação.

[Muitas abordagens](#)

O ER moderno segue uma abordagem de requisitos orientada por valor (consulte o Princípio 1 no Capítulo 2). Consequentemente, o grau em que um requisito atende aos critérios de qualidade fornecidos deve corresponder ao valor criado por esse requisito. Isso tem duas consequências importantes:

- Os requisitos não precisam aderir totalmente a todos os critérios de qualidade.
- Alguns critérios de qualidade são mais importantes do que outros.

[Sem realização universal](#)

Distinguimos entre critérios de qualidade para requisitos únicos e critérios de qualidade para produtos de trabalho de RE, como documentos de RE ou estruturas de documentação.

Para requisitos únicos, recomendamos o uso dos seguintes critérios de qualidade:

[Critérios de qualidade para requisitos únicos](#)

- *Adequado*: o requisito descreve as necessidades verdadeiras e acordadas das partes interessadas.
- *Necessário*: o requisito faz parte do escopo do sistema relevante, o que significa que contribuirá para o alcance de pelo menos um objetivo ou necessidade da parte interessada.
- *Inequívoco*: existe um verdadeiro entendimento compartilhado do requisito, o que significa que todos os envolvidos o interpretam da mesma forma.
- *Completo*: o requisito é autocontido, ou seja, não falta nenhuma parte necessária para sua compreensão.
- *Compreensível*: o requisito é compreensível para o público-alvo, o que significa que o público-alvo pode entender totalmente o requisito.
- *Verificável*: o cumprimento do requisito por um sistema implementado pode ser verificado indiscutivelmente (para que as partes interessadas ou clientes possam decidir se um requisito é ou não cumprido pelo sistema implementado).

Adequação e compreensibilidade são os critérios de qualidade mais importantes. Sem eles, um requisito é inútil ou mesmo prejudicial, independentemente do cumprimento de todos os outros critérios. A verificabilidade é importante quando o sistema implementado deve passar por um procedimento formal de aceitação.

Algumas pessoas usam *correção* em vez de *adequação*. No entanto, a noção de correção implica que existe um procedimento formal para decidir se algo está correto ou não. Como não há procedimento formal para validar um requisito documentado em relação aos desejos e necessidades que as partes interessadas têm em mente, preferimos o termo adequação excesso de correção.

Para produtos de trabalho que cobrem vários requisitos, recomendamos a aplicação dos seguintes critérios de qualidade:

Critérios de qualidade para produtos de trabalho RE

- *Consistente*: não há dois requisitos, registrados em um único produto de trabalho ou em diferentes produtos de trabalho, que se contradigam.
- *Não redundante*: cada requisito é documentado apenas uma vez e não se sobrepõe a outro requisito.
- *Completo*: o produto de trabalho contém todos os requisitos relevantes (requisitos funcionais, requisitos de qualidade e restrições) que são conhecidos neste momento e que estão relacionados a este produto de trabalho.
- *Modificável*: o produto de trabalho é configurado de forma que possa ser modificado sem degradar sua qualidade.
- *Rastreável*: os requisitos no produto de trabalho podem ser rastreados até suas origens, até sua implementação (no design, código e teste) e outros requisitos dos quais dependem.
- *Conformidade*: se houver instruções obrigatórias de estruturação ou formatação, o produto de trabalho deve estar em conformidade com elas.

3.9 Leitura Adicional

Mavin et al. [MWHN2009] apresenta e descreve o modelo EARS. Robertson e Robertson [RoRo2012] descrevem os modelos Volere. Goetz e Rupp [GoRu2003], [Rupp2014] discutem regras e armadilhas para escrever requisitos em linguagem natural.

Cockburn [Cock2001] escreveu um livro inteiro sobre como escrever casos de uso.

Lauesen [Laue2002] discute descrições de tarefas e também fornece alguns exemplos de produtos de trabalho de ER do mundo real.

O padrão ISO/IEC/IEEE 29148 [ISO29148] fornece muitos recursos relativos a produtos de trabalho de ER: modelos de frase, critérios de qualidade para requisitos e descrições detalhadas do conteúdo de vários produtos de trabalho de ER, incluindo um modelo de documento para cada produto de trabalho. Cohn [Cohn2010] tem um capítulo sobre como enquadrar os requisitos em um product backlog.

Gregory [Greg2016] e Glinz [Glin2016] discutem o problema de como requisitos detalhados devem ser especificados e até que ponto especificações de requisitos completas e inequívocas são possíveis.

Numerosas publicações tratam do uso de modelos para especificar requisitos. A especificação UML [OMG2017], bem como livros didáticos sobre UML, descrevem os modelos disponíveis em UML. Hofer e Schwentner [HoSch2020] apresentam modelagem de domínio com storytelling de domínio. [OMG2013] e [OMG2018] descrevem as linguagens de modelagem BPMN para modelagem de processos de negócios e SysML para modelagem de sistemas, respectivamente. Os livros de Booch, Rumbaugh e Jacobson [BoRJ2005], [JaSB2011],

[RuJB2004] fornecem mais profundidade e aplicações (práticas) de UML. Além disso, os seguintes livros e artigos são recomendados para conhecimento e padrões mais aprofundados em requisitos de modelagem: [DaTW2012], [Davi1993], [Fowl1996], [GHJV1994]. [LiSS1994] e [Pohl2010] fornecem uma melhor compreensão dos aspectos de qualidade dos modelos.

4 Práticas para Elaboração de Requisitos

Nos capítulos anteriores, aprendemos sobre a natureza dos requisitos como a representação dos desejos e necessidades das pessoas e organizações por algo novo (por exemplo, um sistema a ser desenvolvido ou adaptado), sobre os princípios que fundamentam a produção dos requisitos, e sobre formas de capturar os requisitos na documentação. Estabelecemos requisitos antes de construir ou modificar um (parte de um) sistema para garantir que o sistema seja útil para — e aceito por — as pessoas ou a organização que o solicitou. Esses requisitos servem como entrada para uma equipe de desenvolvimento que construirá e implementará o sistema.

Isso é Engenharia de Requisitos em poucas palavras; acontece, explícita ou muitas vezes implicitamente, quando e onde quer que as pessoas tentem desenvolver algo. Em princípio, a qualidade dos requisitos determina a qualidade do resultado do desenvolvimento subsequente. Sem requisitos adequados, é improvável que o sistema resultante seja útil. Portanto, é importante elaborar os requisitos de forma profissional. Isso requer uma definição explícita do *como*: as práticas a serem usadas para uma elaboração de alta qualidade.

É disso que trata este capítulo: ele fornece uma visão geral das tarefas, atividades e práticas relevantes para qualquer pessoa envolvida na Engenharia de Requisitos. Começa com a busca de fontes potenciais de requisitos e termina com a entrega de um conjunto de requisitos único, consistente, compreensível e acordado que pode servir como entrada para o desenvolvimento, manutenção e operação eficientes de um sistema eficaz.

Tarefas em ER

A primeira tarefa em todo esforço de Engenharia de Requisitos será identificar e analisar *fontes potenciais de requisitos*. Isso pode parecer uma tarefa simples e óbvia, mas como veremos na Seção 4.1, existem alguns aspectos que precisam ser considerados e analisados. Negligenciar uma fonte inevitavelmente levará a requisitos ruins ou mesmo ausentes e, portanto, degradará a qualidade do sistema resultante.

A próxima etapa é obter os requisitos dessas fontes. É como tirar água de um poço: você nunca sabe o que há no balde até trazê-lo à superfície. Na Engenharia de Requisitos, essa tarefa é chamada de *elicitação*; é explicado na Seção 0. Na elicitação, transformamos desejos implícitos, vontades, necessidades, demandas, expectativas e tudo o mais em requisitos explícitos que podem ser reconhecidos e compreendidos por todas as partes envolvidas.

No entanto, quando você pergunta a duas pessoas sobre seus requisitos para um determinado sistema, raramente obterá exatamente as mesmas respostas. Em toda uma série de requisitos levantados em diferentes fontes, é quase certo que alguns deles serão conflitantes. Como é impossível implementar requisitos conflitantes em um único sistema, a *resolução de conflitos* sempre será uma tarefa importante na Engenharia de Requisitos, conforme descrito na Seção 4.3.

A Seção 4.4 é dedicada à tarefa final da Engenharia de Requisitos: a *validação dos requisitos*. O objetivo desta etapa é garantir que a qualidade do conjunto de requisitos elicitados e dos requisitos individuais dentro desse conjunto seja boa o suficiente para permitir o desenvolvimento subsequente do sistema.

Pela descrição acima das tarefas de Engenharia de Requisitos, você pode ter a impressão de que elas são executadas como um processo linear com uma sequência estrita de etapas.

No entanto, esta certamente não é a intenção desta descrição e raramente é o caso na prática.

A Figura 4.1 mostra algumas etapas do processo que são comuns na Engenharia de Requisitos. Eles podem ser executados em paralelo, em loops ou sequencialmente - o que for adequado em determinada situação.

Não é um processo linear

O ponto de partida geralmente é um conjunto limitado de fontes óbvias. Durante a elicitação dessas fontes, novas fontes são identificadas, desencadeando tarefas de elicitação adicionais. Quando os conflitos são encontrados, uma elicitação mais detalhada pode ser necessária para encontrar uma saída. Na validação, pode parecer que uma fonte foi negligenciada, um requisito está com defeito ou um conflito permaneceu descoberto, resultando em uma nova série de análise de fonte, elicitação e/ou resolução de conflitos e atividades de validação. Mesmo durante o desenvolvimento do sistema subsequente, as circunstâncias podem exigir Engenharia de Requisitos adicional.

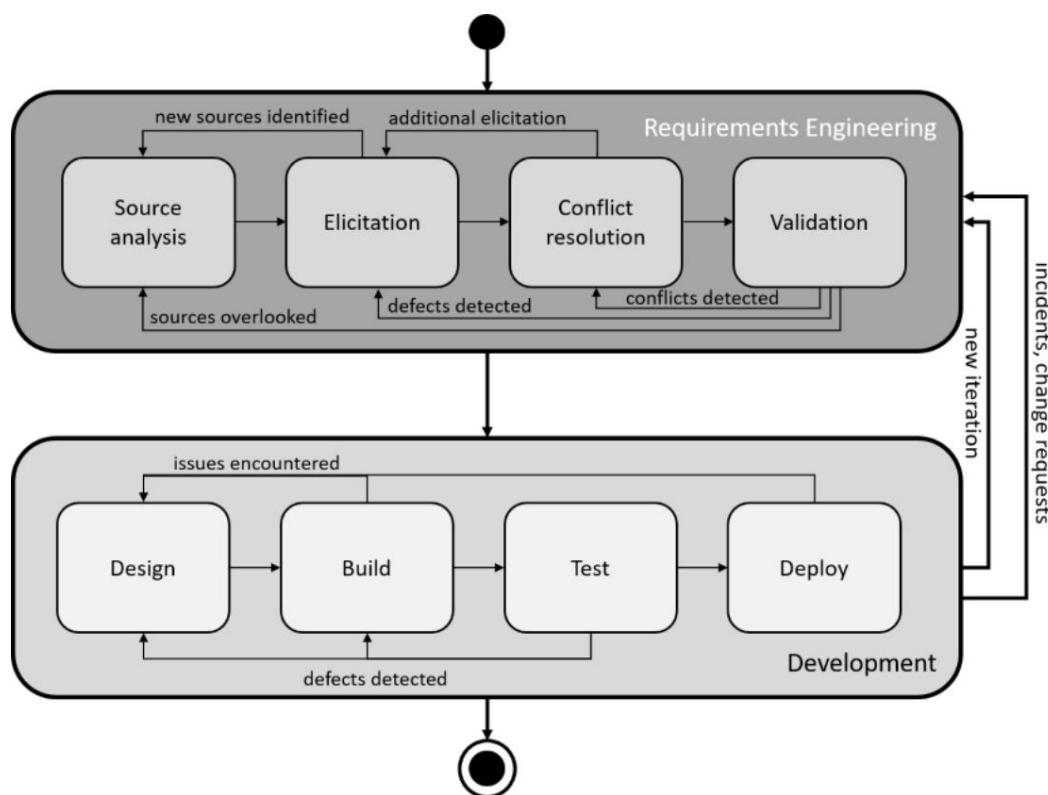


Figura 4.1 A Engenharia de Requisitos não é um processo linear

Em projetos ágeis, a Engenharia de Requisitos iterativa e incremental e o desenvolvimento do sistema andam de mãos dadas, com os requisitos sendo elaborados pouco antes do desenvolvimento de um novo incremento do sistema. Em tais projetos, você frequentemente verá que um projeto começa com um backlog de produto limitado de requisitos de alto nível que são refinados e detalhados apenas quando são candidatos para a próxima iteração.

4.1 Fontes de Requisitos

Os requisitos não são como barras de chocolate, colocadas na prateleira para que todos possam pegá-las como quiserem. Na introdução deste capítulo, compararmos os requisitos com a água a ser retirada de um poço: é um grande esforço trazê-los à superfície.

Portanto, o primeiro problema que um Engenheiro de Requisitos enfrentará é “Onde estão os poços?” Como nenhum requisito vem sem uma fonte, uma das primeiras atividades na elicitação de requisitos é identificar as fontes potenciais. Não basta identificar

essas fontes apenas no início de um projeto ou desenvolvimento de produto; este é um processo que será repetido várias vezes.

Desde o início da elaboração dos requisitos, o Engenheiro de Requisitos deve estar envolvido na identificação, análise e envolvimento de todas as fontes de requisitos relevantes, pois a falta de uma fonte relevante levará inevitavelmente a um entendimento incompleto dos requisitos relevantes. E isso vai continuar até o fim: a identificação das fontes de requisitos é um processo que requer reconsideração constante.

esforço contínuo

O Capítulo 2, Princípio 3 enfatiza a necessidade de (explícito e implícito) *entendimento compartilhado* entre todas as partes envolvidas: stakeholders, Engenheiros de Requisitos, desenvolvedores. Compreender o contexto do sistema a ser desenvolvido em um determinado domínio de aplicação é um pré-requisito para poder identificar as fontes de requisitos relevantes. Conhecimento de domínio, colaboração anterior bem-sucedida, cultura e valores comuns e confiança mútua são facilitadores para o entendimento compartilhado, enquanto distância geográfica, terceirização ou grandes equipes com alta rotatividade são obstáculos.

No Capítulo 2, Princípio 4, introduzimos o contexto como um conceito essencial para entender e especificar um sistema e seus requisitos. Definimos o contexto como aquela parte da realidade que fica entre a *fronteira do sistema* e a *fronteira do contexto*.

As entidades nesse contexto irão de alguma forma influenciar o sistema ou mesmo interagir com ele, mas não estarão contidas no próprio sistema.

Isso tornaria a busca por fontes de requisitos bastante simples: basta olhar ao redor no contexto! Mas não é tão fácil. No início de um processo de desenvolvimento, o contexto ainda não foi definido; o limite do sistema e o limite do contexto ainda precisam ser determinados. Portanto, a busca por fontes de requisitos é um processo iterativo e recursivo.

recursividade

Fontes potenciais são analisadas por sua relação com o futuro sistema. Se você não encontrar nenhuma relação ao analisar uma fonte potencial, isso significa que ela faz parte do ambiente irrelevante e não será analisada quanto aos requisitos. Fontes potenciais que parecem fazer parte do sistema futuro também não interessam ao Engenheiro de Requisitos; eles *pertencem* aos desenvolvedores. Apenas aquelas entidades para as quais a análise revela uma interação, uma interface ou uma influência no sistema futuro, mas que permanecem (relativamente) inalteradas durante o próximo desenvolvimento merecem atenção como fontes de requisitos. Nesta análise, o limite do sistema e o limite do contexto são delineados, vagos a princípio e se tornando mais nítidos à medida que mais e mais fontes são identificadas. À medida que o contexto se torna mais claro, torna-se mais fácil identificar novas fontes, que por sua vez aguçam ainda mais os limites.

A busca por fontes de requisitos geralmente começa com algumas fontes óbvias, muitas vezes identificadas pelo cliente no início de um esforço de desenvolvimento. A elicitação inicial dessas fontes revelará outras fontes potenciais, que serão então analisadas para decidir se são ou não relevantes para o sistema. Durante esta análise, novas fontes potenciais podem aparecer novamente. De fato, em todo esforço de elicitação, o Engenheiro de Requisitos continuará interessado em detectar novas fontes. Isso pode continuar até o final do esforço de desenvolvimento. No entanto, tentamos identificar as fontes principais e mais relevantes com antecedência, porque todas as outras atividades de Engenharia de Requisitos dependem dessa identificação precoce.

Na Engenharia de Requisitos, discernimos três categorias principais de fontes:

- Partes interessadas
- Documentos
- (Outros) sistemas

Essas categorias são discutidas com mais detalhes nas seções a seguir.

4.1.1 Partes Interessadas

No Capítulo 2, Princípio 2, você aprendeu sobre o stakeholder como uma pessoa ou organização que *influencia* os requisitos de um sistema ou é *impactada* por esse sistema.

As partes interessadas de um sistema são as principais fontes de requisitos. Ainda mais do que com outras fontes, a não inclusão de uma parte interessada relevante terá um grande impacto negativo na qualidade do conjunto final de requisitos; descobrir essas partes interessadas tardeamente (ou ignorá-las completamente) pode levar a mudanças caras ou, no final, a um sistema inútil. Para criar um sistema que atenda às necessidades de todas as partes interessadas, a identificação sistemática das partes interessadas deve começar no início de qualquer esforço de desenvolvimento e os resultados devem ser gerenciados ao longo do desenvolvimento. As partes interessadas podem ser encontradas em uma ampla área ao redor do sistema, desde usuários diretos e indiretos do sistema, gerentes (de negócios), equipe de TI, como desenvolvedores e operadores, até oponentes e concorrentes, instituições governamentais e reguladoras e muitos outros. A questão principal para identificar uma pessoa ou organização como parte interessada é: “Existe uma relação relevante entre a pessoa/organização e o sistema?”

Ajuda a ver os stakeholders como seres humanos feitos de carne e osso. Se você identificar uma organização como parte interessada, faça a si mesmo perguntas como as seguintes: “Posso identificar uma pessoa responsável por esta organização? Quem pode ser visto como o contato principal desta organização? Quem representa esta organização dentro da nossa empresa?” Por exemplo, se o governo for a parte interessada porque uma determinada lei está envolvida, procure alguém que represente o governo como a fonte a ser abordada para os requisitos. Neste caso, não é muito útil identificar o Primeiro Ministro como esta pessoa; o chefe do departamento jurídico interno seria uma escolha melhor.

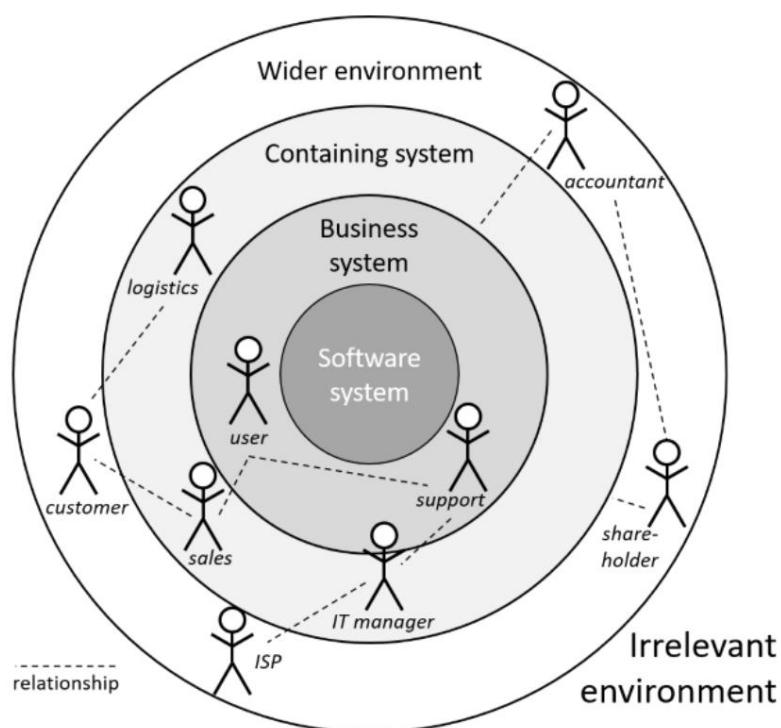


Figura 4.2 Modelo da cebola de Alexander

Não existe uma técnica padrão para identificar as partes interessadas, mas o modelo cebola de Ian Alexander [Alex2005] pode ser um bom começo, veja a Figura 4.2. Este modelo mostra como um sistema (de software) é cercado por várias camadas de sistemas sócio-técnicos¹ e sociais de nível superior, cada um com seus próprios stakeholders. No início de um esforço de desenvolvimento de requisitos, algumas dessas partes interessadas ficarão evidentes - por exemplo, usuários finais ou clientes. Eles podem ser usados como ponto de partida na busca por outras partes interessadas. Depois de identificá-los como fontes relevantes, o Engenheiro de Requisitos analisará seus relacionamentos, tanto nos sistemas circundantes internos quanto externos. Nessa análise, novos stakeholders serão encontrados, que por sua vez poderão ter outras (e mais) relações a serem analisadas. Você poderia chamar isso de princípio da *bola de neve*: quanto mais stakeholders você encontrar, mais fácil será encontrar novos. No entanto, ao chegar às partes interessadas no *ambiente mais amplo de Alexander*, quaisquer relacionamentos externos terminarão no ambiente irrelevante, o que significa que eles não revelarão mais novas fontes.

modelo cebola

Além das partes interessadas referindo-se a outras partes interessadas, os documentos podem frequentemente revelar novas partes interessadas. Bons exemplos são organogramas, descrições de processos, relatórios de marketing e documentos regulamentares. Para obter mais informações sobre documentação como fonte de requisitos, consulte a Seção 4.1.2. As listas de verificação de grupos e papéis típicos de partes interessadas podem ser uma ferramenta útil para evitar a negligência de certas partes interessadas em potencial imperceptíveis. Além disso, analisar as partes interessadas de sistemas legados ou semelhantes pode ajudar.

Como um Engenheiro de Requisitos, você coletará muitos dados sobre suas partes interessadas e manterá esses dados até que seu trabalho seja concluído. Você deve saber quem são as partes interessadas, como alcançá-las, quando e onde estão disponíveis, quais são seus conhecimentos, bem como sua relevância como fonte, qual é sua atitude em relação ao projeto e sua influência sobre ele, qual é sua os papéis estão na empresa, no projeto e em sua relação com o sistema, etc. Normalmente, essas informações são mantidas em uma lista de stakeholders, e devem ser mantidas atualizadas, pois durante todas as etapas, você permanecerá em contato com todas as partes interessadas - algumas intensamente e muito de perto, outras com pouca frequência e superficialmente.

lista de partes interessadas

Veja a Tabela 4.1 abaixo para um exemplo simplificado.

Tabela 4.1 Exemplo de uma lista de partes interessadas

Nome	departamento	Telefone	Disponibilidade	Papel	Influência	Interesse
marlene	Proprietário	482263	segundas-feiras apenas	Patrocinador	++	o
Peter	Vendas	481225	Permanente	Proprietário do produto	++	+
eva	Jurídico	481237	não em junho	Consultor	+	-
Hassan	Logística	242651	Permanente	Do utilizador	o	++
mira	Central de atendimento	242424	depois das 16h	Do utilizador	-	+
Natália*)		481226	Permanente	Cliente	++	++

* Persona, criada, mantida e representada pela equipe *do painel do cliente*

¹ Um sistema sociotécnico é um sistema que considera requisitos que abrangem hardware, software, aspectos pessoais e comunitários, ao mesmo tempo em que reconhece a interação entre as infraestruturas complexas da sociedade e o comportamento humano.

Manter um relacionamento bom e aberto com os stakeholders é fundamental para obter informações relevantes deles. Isso depende principalmente de características comportamentais, como integridade, honestidade e respeito.

Respeito

A comunicação aberta e frequente sobre seus planos, suas atividades e seus resultados é essencial. Você pode ter que transformar as partes interessadas de oponentes iniciais em colaboradores. Como Engenheiro de Requisitos, você deve entender o que as partes interessadas esperam de você. Você também deve *vender* seu trabalho, mostrando a eles os benefícios de sua solução e removendo os impedimentos que as partes interessadas enfrentam ou esperam em seu caminho para essa solução. Infelizmente, não é incomum que certas partes interessadas (principalmente internas) prevejam ou de fato sofram consequências negativas das mudanças que resultam de seu projeto. Nesses casos, será difícil obter a cooperação deles, embora você certamente precise dela. A escalação para níveis mais altos na organização pode ser a única maneira de proceder, mesmo que o relacionamento resultante esteja longe de ser o ideal. O gerenciamento de relacionamento com as partes interessadas [Bour2009] é uma maneira eficaz de combater os problemas com as partes interessadas.

Isso implica um processo contínuo de obter e manter o apoio e o comprometimento das partes interessadas, envolvendo as partes interessadas certas no momento certo e compreendendo e gerenciando suas expectativas.

Para envolver as partes interessadas no processo de elicitação, você deve garantir que elas saibam do que se trata o projeto e qual é o seu papel dentro do projeto. Você deve entender suas necessidades e tentar atendê-las o máximo possível dentro de suas competências no projeto. Embora as partes interessadas devam ser tratadas com respeito, você pode exigir o mesmo das partes interessadas, pelo menos daqueles que estão ativamente envolvidos no projeto. Isso significa que eles devem ter tempo para você quando você precisar deles. Eles devem fornecer as informações solicitadas, bem como outras informações que eles saibam ser relevantes. O feedback deles sobre seus produtos de trabalho deve ser dado em tempo hábil e eles devem evitar fofocas sobre o projeto, etc.

Direitos e deveres

Os problemas com as partes interessadas geralmente surgem se os direitos e obrigações do Engenheiro de Requisitos e das partes interessadas com relação ao sistema proposto ou ao projeto atual não estiverem claros ou se as respectivas necessidades não forem suficientemente atendidas. Se forem encontrados problemas, um tipo de *acordo ou contrato de partes interessadas* pode ajudar a fornecer a clareza desejada a todas as partes. Quando isso ocorre dentro de uma organização, o endosso da alta administração pode contribuir para o sucesso de tal abordagem.

4.1.1.1 Uma parte interessada especial: o usuário

Todo sistema que desenvolvemos terá alguma interação com determinados usuários; por que mais você iria desenvolvê-lo? Claro, quando você está trabalhando nos requisitos para um subsistema técnico embutido, escondido dentro de algum tipo de maquinário complicado, os usuários só irão interagir com ele indiretamente através de várias camadas de sistemas circundantes. Nesses casos, esses usuários não serão suas fontes mais importantes de requisitos. No entanto, em muitos sistemas, seres humanos específicos terão uma interface direta com o sistema: os usuários (finais). A aceitação do sistema é vital para o sucesso do projeto, por isso são de seu interesse primordial e receberão atenção especial durante toda a Engenharia de Requisitos.

Existem duas grandes categorias de usuários:

- Os usuários *internos* estão diretamente relacionados à organização para a qual o sistema está sendo desenvolvido, como funcionários, gerência, subcontratados. Eles são em grande parte limitados em número, mais ou menos conhecidos individualmente e de alguma forma envolvidos no projeto. É relativamente fácil contatá-los e eles podem ser alcançados, influenciados e motivados por meio de canais formais existentes.
- Os usuários *externos* estão fora da empresa, como clientes, parceiros de negócios, civis. O seu número pode ser (muito) grande, em muitos casos não são conhecidos individualmente, e podem ser completamente alheios ou indiferentes ao projeto. Você não pode influenciá-los por meio de canais formais. Para entrar em contato com eles, você pode precisar fazer coisas especiais para motivá-los a participar, como anunciar, prometer alguma recompensa ou dar-lhes acesso gratuito a uma versão beta. Formar um painel de usuários ou abordar a multidão (às vezes com pagamento) pode ser uma maneira útil de envolver esses usuários.

Esteja ciente de que, além dessas categorias regulares, também pode ser relevante prestar atenção aos *usuários indevidos*: pessoas que intencionalmente tentam interagir com o sistema de forma prejudicial, como hackers ou concorrentes. Raramente é possível contatá-los ou influenciá-los, mas você pode tentar desenvolver medidas para desencorajá-los, mantê-los afastados ou detectar tentativas previsíveis de uso indevido.

Esta categorização não deve ser considerada de forma muito estrita. Podemos imaginar projetos em que os usuários de fora da empresa são poucos e podem ser facilmente alcançados, portanto, podem ser vistos como *internos*. Por outro lado, em grandes empresas, a distância até os usuários pode ser tão grande que eles devem ser tratados mais ou menos como *externos*.

Se você tiver uma boa percepção de sua base de usuários, deverá fazer uma distinção entre as funções do usuário. Funções separadas geralmente terão diferentes necessidades de informação, usarão o sistema à sua própria maneira e podem ter direitos de acesso distintos a funções e dados - por exemplo, usuários que inserem dados versus supervisores que verificam essa entrada. Nesses casos, certifique-se de incluir representantes de todas as funções relevantes na elicitação.

[Funções do usuário](#)

Frequentemente, especialmente no início dos esforços de elicitação, esse insight estará ausente. Então, é ainda mais importante perceber que não existe *O Usuário*. O usuário não é uma massa amorfa de humanos idênticos, mas sim uma coleção de indivíduos, cada um com seus próprios hábitos, desejos e necessidades. Quando um sistema tem milhares de usuários ou mais, é claro que você não conseguirá ajustar os requisitos às suas necessidades individuais. Por outro lado, uma abordagem de *tamanho único* para o usuário *médio* também pode não funcionar.

Nesses casos, ajuda a discernir um número de tipos de usuários ou grupos de usuários que mostram certas semelhanças, muitas vezes comportamentais dentro do grupo como distintos de outros grupos. Na prática, geralmente funciona melhor ter de três a sete grupos. Então, como Engenheiro de Requisitos, você tratará cada grupo como uma fonte distinta de requisitos. Uma boa técnica é o uso de *personas* [Hump2017]. Personas são indivíduos fictícios que descrevem grupos típicos de usuários do sistema com necessidades, objetivos, comportamentos ou atitudes semelhantes.

[Grupos de usuários](#)

4.1.1.2 Pessoas

Existem duas abordagens principais para a criação de personagens:

Baseado em dados

Nessa abordagem, as personagens são desenvolvidas com técnicas de marketing, como entrevistas, grupos focais e outras técnicas de coleta de dados etnográficos.

Essas personagens são chamadas de *personas qualitativas*. Se as personagens são desenvolvidas por meio de análise estatística de uma grande amostra de sua base de usuários, elas são chamadas de *personas quantitativas*.

Imaginação

Como alternativa mais barata e rápida, as personagens podem ser desenvolvidas pela imaginação – por exemplo, em uma sessão de brainstorming com a equipe do projeto.

Nós os chamamos de *personas ad hoc* ou *proto-personas*. Como Engenheiro de Requisitos, você deve estar ciente de que personagens ad hoc são baseadas em imaginação e suposições.

Essas suposições devem ser verificadas e confirmadas ao longo do processo de Engenharia de Requisitos.



Figura 4.3 Exemplo de Persona

Basicamente, as descrições de personagens contêm informações relevantes tendo em vista o desenvolvimento do sistema em questão. Normalmente, essas informações serão *enriquecidas* com dados adicionais, como nome, endereço, hobbies e um desenho ou retrato.

Isso dá um rosto humano ao conceito abstrato de personagem. Isso pode ajudá-lo a entender que seu trabalho como Engenheiro de Requisitos não se relaciona apenas com fatos, mas também com emoções. A Figura 4.3 dá um exemplo de descrição de personagem.

Se você usa personagens em seu projeto, pode ser útil procurar alguns indivíduos que se encaixem nas descrições de personagens e tratá-los como representantes de cada grupo. Nesse caso, você tem interessados reais com quem pode se comunicar. No entanto, sempre

Lembre-se que o grupo que tal stakeholder representa é um grupo artificial que não existe no mundo real, mas apenas no contexto do sistema ou projeto.

4.1.2 Documentos

Os documentos também podem ser uma fonte importante de requisitos. Como um Engenheiro de Requisitos, muitas vezes você tem que fazer muitas leituras, especialmente no início de um projeto. Todos os tipos de documentos podem ser relevantes: documentos relacionados à empresa, ao domínio e ao projeto, descrições de produtos e processos, documentação legal e regulamentar, etc. Assim como ocorre com as partes interessadas, você pode fazer uma distinção entre documentos *internos* e *externos*. Documentos internos podem ser fáceis de obter, mas podem ser confidenciais e não podem ser compartilhados sem consentimento explícito.

Freqüentemente, você precisará assinar um acordo de não divulgação antes de receber acesso a eles. Documentos externos podem ser difíceis de encontrar, mas geralmente são públicos; caso contrário, verifique se você tem permissão para acessá-los e usá-los.

Os documentos podem ser uma ótima maneira de encontrar outras fontes. Por exemplo, uma descrição de processo interno pode mencionar certas funções envolvidas nesse processo, o que, por sua vez, pode levar você a novos interessados em potencial. No entanto, os documentos também podem ser fontes diretas de requisitos, especialmente aqueles que são facilmente ignorados ou não mencionados regularmente pelas partes interessadas: restrições em padrões, diretrizes da empresa e outros documentos legais ou regulamentares; requisitos detalhados em procedimentos e instruções de trabalho; novas ideias brilhantes em material de marketing dos concorrentes. Estudar a documentação pode te ajudar a entender o contexto do sistema a ser desenvolvido, até mesmo lendo e-mails entre pessoas que tomaram a iniciativa do projeto.

Ler sobre soluções análogas para problemas e objetivos em outras empresas e domínios pode estimular sua imaginação e mostrar uma direção viável para seu projeto atual.

Como Engenheiro de Requisitos, você deve estar ciente de que um documento está sempre relacionado a algumas pessoas: o(s) autor(es), o público (alvo), um gerente responsável ou um auditor que verifica sua aderência, alguém que apontou sua existência para você, etc

Relacionamento com partes interessadas

Todas essas pessoas podem ter um papel como parte interessada; cabe a você descobrir se é esse o caso ou não. Você deve sempre verificar a validade e a relevância de um documento e geralmente precisa que as partes interessadas confirmem isso para você. Se você derivasse requisitos de um documento inválido ou desatualizado, o sistema desenvolvido a partir desses requisitos provavelmente falharia.

Assim como as partes interessadas, os documentos usados como fontes de requisitos devem ser gerenciados. Você pode usar uma lista de documentos, comparável à lista de partes interessadas discutida acima. Todos os documentos devem ser mantidos em algum tipo de biblioteca indexada comum com uma identificação única para permitir que sejam referenciados. Datas e números de versão são importantes para evitar o trabalho com versões desatualizadas; você pode verificar em intervalos regulares se uma versão mais recente foi publicada e se isso influencia os requisitos. Você deve trabalhar preferencialmente com versões finais, mas na prática, muitas vezes você tem que lidar com rascunhos, então você também tem que registrar o status dos documentos. Mantenha as versões antigas em um arquivo, porque elas podem ser importantes para entender como um sistema e seus requisitos evoluíram. Configurar o gerenciamento adequado e preciso dos documentos envolvidos desde o início de um projeto economizará muito trabalho posteriormente, em Engenharia de Requisitos, desenvolvimento e implantação. É um bom ponto de partida para estabelecer a rastreabilidade reversa, conforme discutido na Seção 6.6.

4.1.3 Outros Sistemas

Você também pode considerar outros sistemas como fontes de requisitos do sistema em que está interessado. Aqui, você pode fazer uma distinção entre sistemas *internos* e *externos*, assim como na documentação e com as mesmas considerações sobre acesso e confidencialidade. Outra distinção é a de sistemas *semelhantes* versus sistemas *de interface*.

Sistemas semelhantes possuem certas funcionalidades em comum com o sistema a ser desenvolvido. Eles podem ser sistemas predecessores ou legados, sistemas concorrentes, sistemas comparáveis usados em outras organizações, etc. Você frequentemente os estuda por meio de sua documentação, mas às vezes pode observá-los em ação ou experimentá-los como se fossem uma espécie de protótipo. Você pode entrar em contato com seus usuários para saber mais sobre as funcionalidades e soluções de tais sistemas. Sistemas predecessores ou legados geralmente são uma boa fonte para identificar requisitos e restrições detalhados (funcionais e de qualidade).

Sistemas semelhantes

No entanto, esteja ciente de que as restrições (especialmente técnicas) do passado podem não ser mais relevantes e não podem mais restringir seu espaço de solução atual.

Às vezes, um novo sistema e um sistema legado irão coexistir durante um determinado período, levando a requisitos adicionais – por exemplo, com relação ao compartilhamento de dados. Os sistemas concorrentes e comparáveis podem ser estudados por suas características de solução e podem ser uma boa fonte para identificar *encantadores* (consulte a Seção 4.2.1).

Os sistemas de interface terão um relacionamento direto com o sistema para o qual você está desenvolvendo os requisitos. Eles trocarão dados com seu sistema como uma fonte e/ou um coletor por meio de alguma interface (síncrona ou assíncrona, em tempo real ou em lote) (consulte também a Seção 3.4.2 sobre interfaces do sistema na modelagem de contexto). A configuração, o conteúdo e o comportamento corretos de tal interface geralmente são essenciais para garantir que seu sistema funcione e, portanto, você terá que entender o sistema em detalhes. Você pode estudar sistemas de interface por meio de sua documentação, mas como todos os detalhes (técnicos) são importantes aqui, simulação ou teste podem ser necessários.

Sistemas de interface

No que diz respeito a sistemas mais antigos ou legados em particular, você não pode confiar que sua documentação esteja atualizada, então você precisará de alguma prova. Para entender uma interface, você também terá que entender o contexto, a funcionalidade e o comportamento do sistema de interface. Será útil se você puder entrar em contato com gerentes de aplicativos, arquitetos ou designers de tais sistemas, especialmente se o próprio sistema de interface precisar ser atualizado para permitir a nova interface. Esteja também ciente de que um sistema de interface terá usuários; pode ser interessante considerar esses usuários como partes interessadas no *ambiente mais amplo* de Alexander de seu próprio sistema.

4.2 Elicitação de Requisitos

Se continuarmos a analogia da água sendo retirada de um poço, chegamos ao ponto em que encontramos o poço e começamos a puxar a corda para encher o balde com a água necessária (ou neste caso requisitos) para a superfície. Isso é o que chamamos de *elicitação*: o esforço despendido pelo Engenheiro de Requisitos para transformar desejos implícitos, demandas, desejos, necessidades, expectativas – que até agora estavam ocultos em suas fontes – em requisitos explícitos, compreensíveis, reconhecíveis e verificáveis. Claro, teremos que usar todos os poços para ficar completo e puxar a corda da maneira certa para garantir que toda a água chegue à superfície. Na terminologia da Engenharia de Requisitos, dizemos que devemos aplicar as *técnicas corretas de elicitação*.

Uma categorização comum das técnicas de elicitação é a distinção entre:

- técnicas de coleta

► Técnicas de design e geração de ideias

A partir dessas categorias, você pode selecionar uma ampla gama de técnicas de elicitação, cada uma com suas próprias características. A Figura 4.4 oferece uma visão geral das técnicas de elicitação em suas categorias e subcategorias.

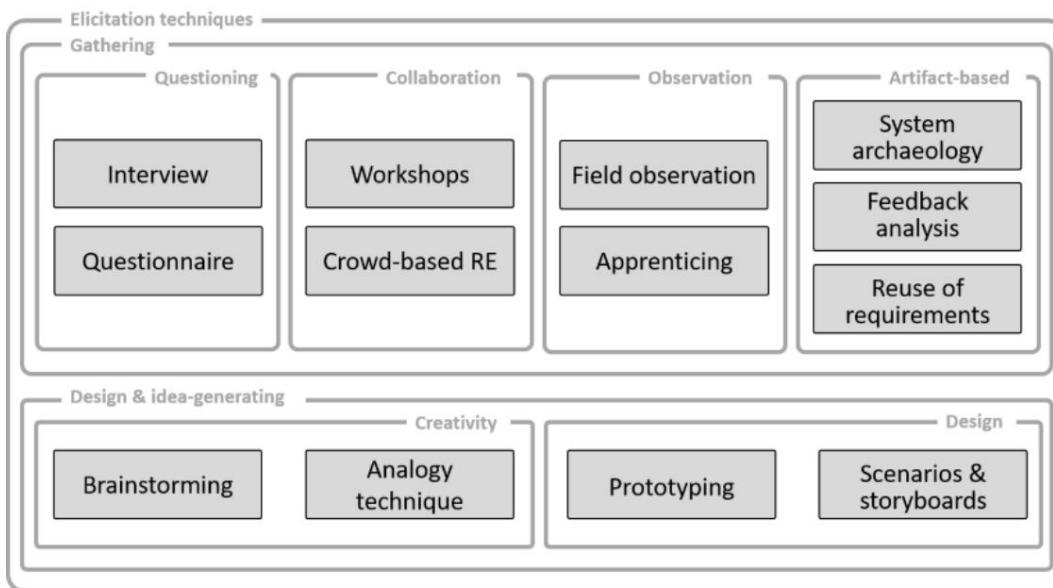


Figura 4.4 Uma visão geral das técnicas de elicitação

Uma competência chave crítica do Engenheiro de Requisitos é a habilidade de escolher as técnicas corretas (combinação de) sob determinadas circunstâncias. Escolher os corretos pode depender de muitos fatores, como:

► Tipo de sistema

Um sistema inovador completamente novo se beneficiará mais das técnicas de design e geração de ideias, enquanto um sistema de substituição em um ambiente crítico de segurança pode precisar de técnicas de questionamento e arqueologia do sistema.

Selecionando a técnica certa

► Modelo de ciclo de vida de desenvolvimento de software

Em um projeto em cascata, você pode ter planejado técnicas extensas, como aprendizado ou analogias, enquanto em um ambiente ágil, brainstorming, storyboard e prototipagem podem prevalecer.

► Pessoas envolvidas

Por exemplo, a observação de campo provavelmente não será apreciada em negócios altamente confidenciais; uma pesquisa abrangente pode ser preferida a um grande número de entrevistas individuais.

► Configuração organizacional

Uma organização governamental sólida precisa de uma abordagem totalmente diferente de uma jovem startup; uma empresa dispersa e altamente descentralizada precisa de uma abordagem diferente de uma empresa compacta com um único local.

Os melhores resultados são geralmente alcançados com uma combinação de diferentes técnicas de elicitação. Para uma abordagem sistemática para selecioná-las, consulte [CaDJ2014].

As técnicas de elicitação são – ou pelo menos deveriam ser – capazes de detectar todos os tipos de requisitos. Na prática da Engenharia de Requisitos, no entanto, requisitos *funcionais* explícitos

Requisitos e restrições de qualidade

os *requisitos* são frequentemente superestimados e os *requisitos* e *restrições* de qualidade mais implícitos recebem menos atenção.

Isso pode resultar em um sistema que - com todos os requisitos funcionais atendidos - não funciona, tem usabilidade ruim, não cumpre as diretrizes de arquitetura ou falha em atender a outros requisitos ou restrições de qualidade e, consequentemente, não será aceito.

As partes interessadas podem ser fontes, mas muitas vezes você encontrará mais informações em documentos. Para a elicitação de *requisitos de qualidade*, a aplicação de uma lista de verificação com base no padrão ISO 25010 [ISO25010] pode ajudar a detectá-los e quantificá-los - por exemplo, na preparação para uma entrevista. As *restrições* podem ser encontradas considerando possíveis restrições do espaço da solução - por exemplo, questões técnicas, arquitetônicas, legais, organizacionais, culturais ou ambientais. Muitas vezes, a documentação relevante pode ser identificada por meio de membros da equipe.

4.2.1 O Modelo Kano

Uma das principais circunstâncias a considerar na seleção de uma técnica de elicitação é a natureza e a importância de um requisito que estamos tentando descobrir. Para obter mais informações sobre a natureza de certos requisitos, o modelo Kano [Verd2014] é útil. Este modelo, mostrado na Figura 4.5, classifica as características de um sistema em três categorias:

modelo Kano

- **Encantadores** (sinônimos: fatores de excitação, requisitos inconscientes)

Um prazer é um recurso que os clientes não conhecem; é por isso que os chamamos de *inconscientes*. Os clientes não pedem o recurso porque não sabem que é possível no sistema – por exemplo, um smartphone que pode ser transformado em projetor. A princípio, quando o recurso é novo no mercado, a maioria dos clientes terá dúvidas sobre ele, mas quando alguns dos primeiros usuários o experimentaram e começaram a divulgá-lo, mais e mais pessoas o desejaram. Se um encantador estiver ausente, ninguém reclamará; mas quando presente, pode ser um diferencial que atrai muitos clientes.

- **Satisfatores** (sinônimos: fatores de desempenho, requisitos conscientes)

Um satisfator é algo que os clientes pedem explicitamente (portanto, requisitos *conscientes*). Quanto mais satisfatórios você puder colocar em seu sistema, maior será a satisfação dos clientes. Um exemplo pode ser o número de lentes e opções de vídeo em um smartphone moderno. Como adicionar recursos satisfatórios geralmente também significa custos mais altos, muitas vezes você precisará de um tipo de análise de custo/benefício para decidir quantos deles serão incorporados ao sistema.

- **Insatisfatórios** (sinônimos: fatores básicos, requisitos subconscientes)

Um insatisfatório também é uma característica que os clientes não pedem. Aqui, no entanto, a razão para não pedir é que o recurso é tão óbvio (*subconsciente*) que os clientes não conseguem imaginar que não faça parte do sistema; esses recursos são tacitamente considerados obrigatórios. Imagine um smartphone sem GPS. Se um insatisfatório for incluído como uma característica de um sistema, os clientes não o notarão porque acham que o sistema não pode existir sem ele. No entanto, se você ignorar esse requisito e deixá-lo fora do sistema, os clientes ficarão muito chateados e se recusarão a usar o sistema.

O modelo Kano analisa os requisitos do ponto de vista do cliente. Centra-se em características diferenciadoras, em oposição às necessidades expressas. Com o modelo Kano em mente, você pode encontrar mais requisitos do que quando se concentra apenas nas necessidades explicitamente formuladas pelas partes interessadas. Como veremos mais adiante neste capítulo, todas as categorias podem ser vinculadas a diferentes técnicas de elicitação.

Perspectiva do cliente

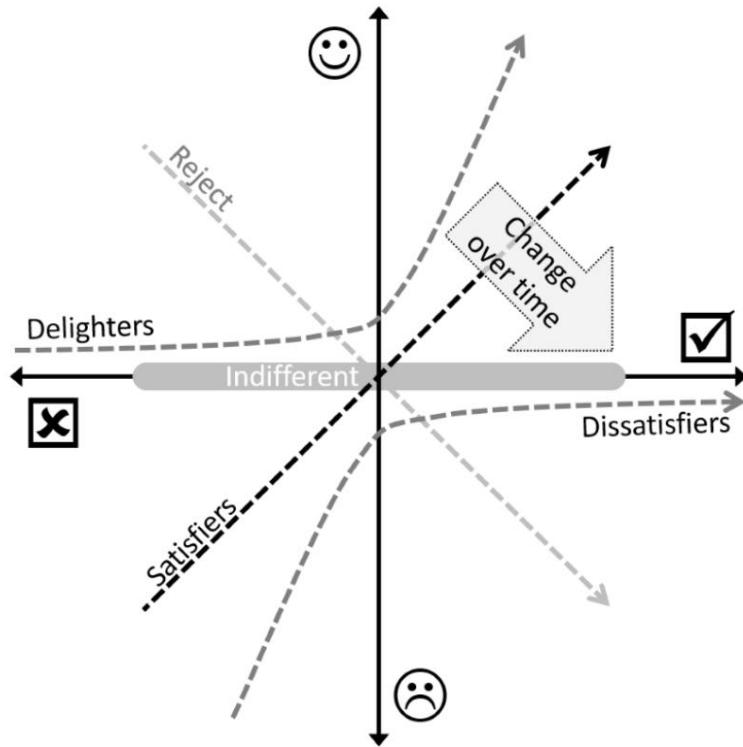


Figura 4.5 O modelo de Kano

Na verdade, o modelo Kano original contém mais duas categorias, os requisitos indiferentes (ou não me importo) e os requisitos rejeitados (ou odeio). Essas categorias não recebem muita atenção na maioria dos manuais de Engenharia de Requisitos, mas ainda podem ser úteis para você como Engenheiro de Requisitos. Suponha, por exemplo, que os desenvolvedores desejem adicionar um determinado recurso ao sistema por motivos técnicos. Se, após análise, verificar que os clientes são indiferentes a esta funcionalidade, é seguro incluí-la no sistema. No entanto, se for um requisito de rejeição, você deve informar aos desenvolvedores que procurem uma alternativa menos prejudicial, pois a implementação desse requisito pode ser um erro caro.

Uma observação interessante ao trabalhar com o modelo Kano é que os requisitos tendem a mudar com o tempo. Se alguém introduzir um novo recurso, não há certeza sobre como o mercado reagirá a esse recurso. Às vezes, os clientes serão indiferentes a ele e o recurso sobreviverá apenas se não aumentar o preço do produto.

Se os clientes o rejeitarem, o recurso provavelmente será removido do produto o mais rápido possível. No entanto, quando (talvez inicialmente uma vanguarda de) os clientes gostarem do recurso, ele se tornará um prazer, um ponto de venda exclusivo pelo qual os clientes estão dispostos a pagar o preço. À medida que mais e mais clientes descobrem, experimentam e gostam desse novo recurso, ele se tornará uma satisfação explicitamente solicitada. Gradualmente, quando sistemas semelhantes começarem a implementar o mesmo recurso, os clientes podem esquecer que

os sistemas originalmente não incluíam tal recurso e o tomarão como certo, tornando-o insatisfatório. É por isso que muitos sistemas contêm recursos que os usuários consideram indispensáveis sem saber por que e, portanto, sem solicitá-los explicitamente.

Um bom exemplo é a função de câmera dos celulares, cujo processo demorou menos de 20 anos. A primeira vez que uma câmera foi introduzida como parte de um telefone celular, a maioria dos clientes ficou perplexa: ninguém havia solicitado esse recurso e a maioria dos clientes pensou: "Se eu quiser tirar uma foto, preciso de uma câmera". No entanto, alguns dos primeiros usuários experimentaram e descobriram a conveniência de tirar fotos sem uma câmera dedicada e poder compartilhá-las instantaneamente com outras pessoas sem imprimir. Eles gostaram do recurso da câmera como um deleite e todas as marcas começaram a implementá-lo em seus telefones, tornando-o mais satisfatório: quanto melhores as fotos, mais satisfeito o usuário fica.

Hoje em dia, na hora de comprar um celular novo, todo mundo dá por certo que ele vai ter a função de câmera, então isso se tornou um insatisfatório: "Se eu não conseguir tirar foto com esse celular, não adianta".

Como você pode categorizar um recurso específico? Você usa a técnica de análise de Kano. Para um recurso específico, você faz duas perguntas a um grupo representativo de usuários em potencial: (1) "O que você sentiria se esse recurso estivesse presente no sistema?" e (2) "O que você sentiria se esse recurso estivesse ausente do sistema?" Você os deixa pontuar as respostas em uma escala de 5 pontos entre "eu amo isso" e "eu odeio isso" e, em seguida, plota a resposta média na matriz de análise de Kano, conforme mostrado na Figura 4.5. A célula que aparece fornece a classificação Kano para o recurso.

matriz de análise de Kano

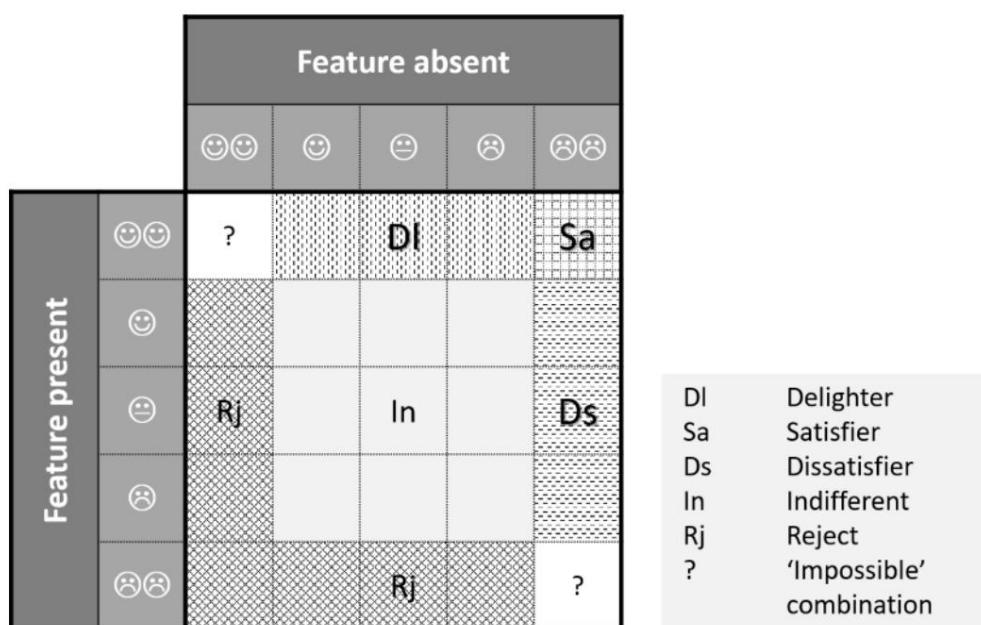


Figura 4.6 Matriz de análise de Kano

A próxima pergunta é: por que se preocupar com a análise Kano na elicitação de requisitos?

Conforme explicamos nas seções a seguir, você precisará de diferentes técnicas para encontrar essas diferentes categorias de recursos. Por si só, as partes interessadas falarão principalmente sobre seus satisfatores - seus requisitos conscientes que solicitam explicitamente. É muito mais difícil detectar as outras categorias, mas, felizmente, existem várias técnicas úteis para fazer isso.

4.2.2 Técnicas de Coleta

Com as *técnicas de coleta*, você examina as diferentes fontes que identificou e extrai os requisitos de lá. Essas técnicas estabelecidas têm sido comumente usadas em toda a Engenharia de Requisitos e produzem predominantemente satisfatórios e insatisfatórios.

As técnicas de coleta podem ser subdivididas em quatro categorias:

- Técnicas de questionamento
- Técnicas de colaboração
- técnicas de observação
- Técnicas baseadas em artefatos

As *técnicas de questionamento* são sempre utilizadas na interação com os stakeholders. O Engenheiro de Requisitos faz perguntas apropriadas às partes interessadas, a fim de permitir que as partes interessadas pensem e recebam respostas das quais os requisitos podem ser derivados.

Técnicas de questionamento

Exemplos de técnicas de questionamento são:

- *entrevistas*

Devido à sua flexibilidade, as *entrevistas* são provavelmente uma das técnicas de elicitação mais utilizadas. Eles não requerem ferramentas específicas e podem ser usados para eliciar requisitos de alto nível, bem como requisitos muito específicos. Normalmente, uma entrevista é uma sessão individual entre um Engenheiro de Requisitos (entrevistador) e uma parte interessada individual (entrevistado), mas um pequeno grupo de entrevistados também é uma opção. Normalmente, os requisitos eliciados com uma entrevista são satisfatórios, pois o entrevistado expressa informações conscientes. A técnica de entrevista não é excessivamente complicada e a maioria das pessoas tem uma boa compreensão do que é. No entanto, você precisa de objetivos claros e boa preparação para obter resultados úteis. As entrevistas podem revelar informações detalhadas e oferecer flexibilidade com base nas respostas dadas. Eles consomem bastante tempo, portanto, essa técnica é menos apropriada quando você deseja atingir um grande número de partes interessadas.

- *Questionários*

Com um *questionário*, um grupo maior de partes interessadas é solicitado a responder — oralmente, por escrito ou em uma página da Web — ao mesmo conjunto de perguntas, que são apresentadas de forma estruturada. Questionários quantitativos são usados para confirmar hipóteses ou requisitos previamente levantados. Eles usam perguntas fechadas (apenas respostas predefinidas são permitidas) e podem, portanto, ser avaliados rapidamente e fornecer informações estatísticas. Por outro lado, os questionários qualitativos usam perguntas abertas e podem encontrar novos requisitos. Eles tendem a fornecer resultados complexos e, portanto, geralmente consomem mais tempo para preparar e avaliar. Em geral, os questionários são uma técnica preferida para grandes grupos. Esteja ciente, no entanto, que projetar um bom questionário envolve muito esforço. Um questionário costuma ser o próximo passo após a obtenção de uma ideia preliminar com base em uma série de entrevistas para validar essas ideias em um grupo maior.

Na categoria de *técnicas de colaboração*, encontramos todos os tipos de colaboração entre o Engenheiro de Requisitos e outras pessoas (stakeholders, especialistas, usuários, clientes, etc.). Alguns exemplos são:

Técnicas de colaboração

- oficinas

Workshop é um termo abrangente para técnicas orientadas a grupos, variando de pequenas reuniões informais a eventos organizados com várias dezenas ou mesmo centenas de partes interessadas. Uma boa definição é a seguinte: "Um workshop de requisitos é uma reunião estruturada na qual um grupo cuidadosamente selecionado de partes interessadas e especialistas em conteúdo trabalham juntos para definir, criar, refinar e concluir as entregas (como modelos e documentos) que representam requisitos do usuário" [Gott2002]. Com um workshop, você consegue ter um bom insight global em pouco tempo, pois utiliza a interação entre os participantes. Se você precisar de mais detalhes, entrevistas ou questionários de acompanhamento são apropriados. As oficinas podem servir como uma técnica de reunião, mas também podem ser usadas em técnicas de criatividade (ver Seção 4.2.3).

- Engenharia de requisitos baseada em multidão

Na Engenharia de Requisitos *baseada em multidão* (também conhecida como baseada em plataforma) (consulte [GreA2017]), a elicitação é transformada em um esforço participativo com uma multidão de partes interessadas, em particular os usuários, levando a requisitos mais precisos e em última análise, um software melhor. O poder da multidão reside na diversidade de talentos e conhecimentos disponíveis dentro da multidão. Como a quantidade de dados obtidos da multidão será grande, uma plataforma automatizada para o processamento desses dados é essencial. Essa plataforma deve oferecer recursos orientados à comunidade que apoiam a colaboração e o compartilhamento de conhecimento e promovem o envolvimento de grupos maiores de partes interessadas na coleta, análise e desenvolvimento de requisitos de software, bem como validação e priorização desses requisitos em um ambiente dinâmico e orientado ao usuário.

Técnicas de observação também são aplicadas em relação aos stakeholders. As partes interessadas são observadas enquanto estão envolvidas em seus processos normais (negócios) em seu contexto usual, sem interferência direta do Engenheiro de Requisitos.

técnicas de observação

As técnicas de observação são particularmente úteis para identificar os insatisfeitos. Você pode observar atividades peculiares, seqüências, dados, etc. que são tão comuns aos envolvidos que eles não os mencionam, e esses aspectos, portanto, não vêm à tona facilmente nas técnicas de coleta.

Formas comuns de técnicas de observação são:

- observação de campo

Durante a *observação de campo*, o Engenheiro de Requisitos observa (principalmente) os usuários finais em seu ambiente enquanto esses usuários executam as atividades para as quais um sistema será desenvolvido. A observação de campo é normalmente usada em situações em que a interação distrairia os usuários ou interferiria no processamento e potencialmente falsificaria os resultados. Pode até ser aplicado sem informar os sujeitos observados, por exemplo, sentando-se com outros pacientes na sala de espera de um dentista para observar seu comportamento. Com a observação de campo, você será capaz de detectar requisitos (muitas vezes detalhados) que não seriam facilmente

encontrado com outras técnicas - por exemplo, porque ações e comportamentos são muito complicados para colocar em palavras.

Esteja ciente de que a observação de campo requer muita preparação, um olhar atento e muito tempo. O vídeo é bastante útil para capturar o comportamento das partes interessadas. Ele pode ser usado em conjunto com a observação direta de campo e pode até substituí-lo em situações onde a presença real do Engenheiro de Requisitos não é permitida ou desejada. O vídeo oferece a possibilidade de pós-processamento para permitir a investigação detalhada de atos e processos de difícil observação.

aprendiz

A *aprendizagem* difere da observação de campo por ser participativa. No estágio, o Engenheiro de Requisitos (*aprendiz*) faz uma espécie de estágio no ambiente em que o sistema em questão será utilizado (ou já está em uso) e usuários experientes (mestres) ensinam ao aprendiz como as coisas funcionam. O aprendiz participa, mas não interfere; eles agem como um novato no campo e podem cometer erros e perguntar “burro”

questões. O objetivo é criar uma compreensão profunda do domínio, do negócio e dos processos antes do início da elicitação real dos requisitos. Frequentemente, será necessário um acompanhamento com entrevistas e questionários para verificar as ideias iniciais. Os requisitos resultantes podem ser posteriormente documentados e validados. A duração ideal para tal estágio depende de muitos fatores diferentes (por exemplo, complexidade do processo, repetitividade, disponibilidade de tempo do mestre e do aprendiz), mas geralmente varia entre um dia e várias semanas. Esteja ciente de que o aprendizado pode ser difícil ou impossível de organizar em certos domínios, como medicina, aviação ou militar.

As *técnicas baseadas em artefatos* não usam partes interessadas (diretamente), mas sim produtos de trabalho, como documentos e sistemas, ou mesmo imagens, arquivos de áudio e vídeo, como fontes de requisitos. Essas técnicas podem encontrar (às vezes muito detalhadamente) satisfatórios e insatisfatórios. Geralmente, é uma tarefa demorada examinar detalhadamente os produtos de trabalho (geralmente mal estruturados, desatualizados ou parcialmente irrelevantes). No entanto, as técnicas baseadas em artefatos podem ser úteis, principalmente quando as partes interessadas não estão prontamente disponíveis.

[Técnicas baseadas em artefatos](#)

Alguns exemplos de técnicas baseadas em artefatos são:

[Exemplos de técnicas baseadas em artefatos](#)

Arqueologia do sistema

Na *arqueologia de sistemas*, os requisitos são extraídos de sistemas existentes – como sistemas legados, sistemas concorrentes ou mesmo sistemas análogos – analisando sua documentação (projetos, manuais) ou implementação (código, comentários, scripts, histórias de usuários, casos de teste). Essa técnica é usada principalmente se um sistema existente tiver sido usado por muitos anos e agora for substituído por um novo sistema por qualquer motivo; o novo sistema deve cobrir a mesma funcionalidade do antigo, ou pelo menos algumas partes dele. A arqueologia do sistema geralmente leva muito tempo, mas pode revelar requisitos e restrições detalhados que não são facilmente detectados de outra forma. No entanto, você precisará de mais tempo para verificar, por meio de outros canais, se esses requisitos ainda são válidos e relevantes.

➤ Análise de feedback

Existem muitas maneiras de coletar *feedback* de (potenciais) usuários e clientes, seja em um sistema existente ou em um protótipo. Os dados de feedback podem ser estruturados (por exemplo, uma classificação de 5 estrelas em uma loja de aplicativos) ou não estruturados (como comentários de revisão). Pode ser coletado por meio de pesquisas na web e formulários de contato, durante testes beta ou A/B, nas mídias sociais ou até mesmo como comentários de clientes recebidos em uma central de atendimento. Muitas vezes, a quantidade de dados é muito grande e a análise será demorada. No entanto, o feedback pode ser muito útil para obter informações sobre as *dores e os ganhos do usuário*.

Pontuações negativas e comentários críticos irão ajudá-lo a detectar insatisfatórios despercebidos. Pontuações positivas e elogios fornecerão informações adicionais sobre os satisfatórios. Ocasionalmente, os comentários podem até conter ideias inovadoras que podem ser transformadas em prazeres. A análise de feedback pode, portanto, resultar no ajuste de requisitos existentes, mas também na descoberta de novos.

➤ Reutilização de requisitos

Muitas organizações já possuem uma grande coleção de requisitos que foram levantados e elaborados no passado para sistemas anteriores. Muitos desses requisitos também podem ser aplicáveis a um novo sistema, especialmente requisitos derivados de um modelo de domínio abrangente.

Portanto, a *reutilização* de requisitos existentes pode economizar muito tempo e dinheiro porque você pode ignorar sua elicitação. No entanto, isso funciona apenas se essa coleção de requisitos existentes estiver atualizada, gerenciada de forma eficaz, facilmente disponível e documentada extensivamente, o que, infelizmente, não costuma ser o caso. Mesmo que a reutilização seja viável, saiba que ainda é preciso validar com os stakeholders se esses requisitos reutilizáveis são relevantes e válidos na nova situação, seja diretamente ou com alguns ajustes.

4.2.3 Técnicas de Design e Geração de Ideias

No passado, a Engenharia de Requisitos concentrava-se em reunir e documentar os requisitos necessários de todas as partes interessadas relevantes, aplicando as técnicas de coleta apresentadas na seção anterior. A crescente influência do software como um direcionador de inovação em muitos negócios agora está exigindo cada vez mais um novo posicionamento da Engenharia de Requisitos como uma atividade criativa de solução de problemas. Isso envolve a aplicação de outras técnicas que não consideram mais as partes interessadas (e seus documentos e sistemas) a única fonte de requisitos.

Técnicas de design e geração de ideias

Sistemas inovadores precisam de recursos novos, talvez disruptivos, que os stakeholders atuais não podem imaginar (ainda).

Técnicas de design e geração de ideias surgiram para atender a essa necessidade. Essas técnicas visam estimular a criatividade, principalmente em equipes, para a geração de ideias e podem fornecer formas adicionais de elaboração de uma determinada ideia. Essas técnicas podem produzir requisitos novos e inovadores que muitas vezes são prazerosos.

Existem muitas técnicas diversas dentro desta ampla categoria, algumas notavelmente simples, outras bastante elaboradas. Veremos alguns exemplos de duas subcategorias:

- técnicas de criatividade
- técnicas de design

Além disso, veremos o campo emergente do *design thinking*.

As técnicas de criatividade estimulam a criatividade para encontrar ou criar novos requisitos que não podem ser coletados diretamente dos stakeholders porque os stakeholders não estão cientes da viabilidade de certos novos recursos ou inovações (técnicas). Essas técnicas são geralmente aplicadas em diversas equipes multidisciplinares de equipe de TI, como analistas, engenheiros de requisitos, desenvolvedores, testadores, proprietários de produtos, gerentes de aplicativos, etc., com ou sem representantes de negócios, usuários, clientes e outras partes interessadas. As técnicas estimulam o pensamento inovador e sem fronteiras e a elaboração das ideias uns dos outros. Infelizmente, nenhum deles garante o sucesso na geração de resultados criativos, pois vários mecanismos em nosso cérebro precisam se unir para possibilitar ideias criativas.

[técnicas de criatividade](#)

Um exemplo óbvio em que as técnicas de criatividade são importantes é a indústria de jogos. É claro que você pode perguntar aos jogadores sobre seus requisitos com uma técnica de coleta e aprenderá o que os jogadores gostam ou não nos jogos atuais. No entanto, para desenvolver um jogo de sucesso, você precisa surpreender os jogadores com algo novo; você tem que descobrir seus deleites. É exatamente aí que as técnicas de criatividade se encaixam.

Várias pré-condições foram identificadas como fatores importantes para o desenvolvimento da criatividade.

[Pré-condições](#)

- Chance – e, portanto, tempo – para uma ideia surgir
- Conhecimento do assunto, o que aumenta as chances de uma ideia que faz a diferença
- Motivação, pois nosso cérebro só pode ser criativo se houver um benefício direto para sua proprietário
- Segurança e proteção, pois ideias inúteis não devem ter consequências negativas

Dois exemplos de técnicas de criatividade são apresentados aqui:

[Exemplos de técnicas de criatividade](#)

➤ Debate

Brainstorming (ver [Osbo1948]) apoia o desenvolvimento de novas ideias para uma determinada questão ou problema. Como acontece com a maioria das técnicas de criatividade, o ponto crucial do brainstorming é adiar o julgamento, separando a descoberta de ideias da análise de ideias.

Algumas diretrizes gerais para o brainstorming incluem: o A quantidade prevalece sobre a qualidade.

o Livre associação e pensamento visionário são explicitamente desejados. o Aceitar e combinar ideias expressas é permitido e desejado. o É proibido criticar as ideias de outros participantes, mesmo que uma ideia

parece ser um absurdo.

o Após uma sessão de brainstorming, as ideias que surgiram são categorizados, avaliados e priorizados. As ideias selecionadas servem como entrada para posterior elicitação.

➤ Técnica de analogia A

técnica de analogia (ver [Robe2001]) ajuda no desenvolvimento de ideias para tópicos críticos e complexos. Ele usa analogias para apoiar o pensamento e a geração de ideias. Seu sucesso ou fracasso é influenciado principalmente pela seleção de uma analogia adequada para o problema em questão. A analogia selecionada pode estar próxima (por exemplo, o mesmo problema em outro negócio) ou distante (por exemplo,

comparando uma organização com um organismo vivo) o problema original. A aplicação da técnica de analogia consiste em duas etapas:

- o Elaborar detalhadamente os aspectos da analogia selecionada sem referindo-se ao problema original.
- o Transferir todos os aspectos identificados da analogia de volta ao original problema.
- o Os conceitos e ideias resultantes serão um ponto de partida para elicitação adicional.

As técnicas de design ajudam a explorar e elaborar ideias geradas com técnicas de criatividade e também ajudam a esclarecer e concretizar necessidades vagas das partes interessadas. Eles dependem fortemente de artefatos visuais ou tangíveis, cooperação em equipe e feedback do cliente.

técnicas de design

Técnicas populares nesta categoria incluem:

➤ Prototipagem

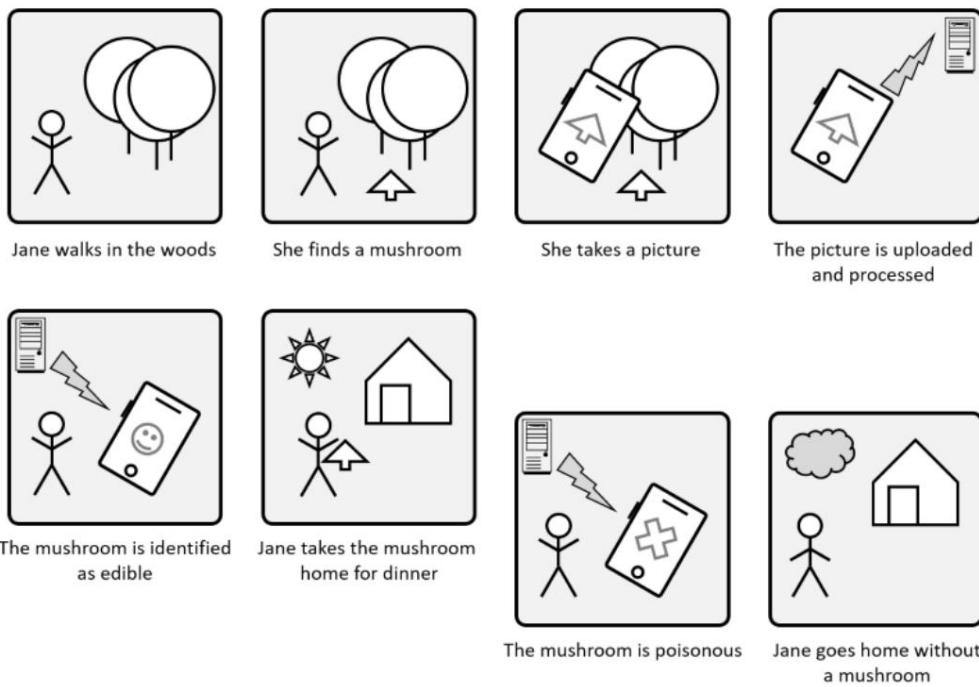
Por *protótipo* (em relação à elicitação; consulte também a Seção 3.7 para obter mais informações), queremos dizer um tipo de produto de trabalho intermediário que é criado ou liberado para gerar feedback. Os protótipos podem variar de simples rascunhos em papel a versões funcionais de pré-lançamento de um sistema. Eles permitem que futuros usuários experimentem o sistema de forma mais ou menos tangível e investiguem certas características, ainda não claras, durante a Engenharia de Requisitos e antes da implementação real. Como veremos na seção sobre validação (4.4.2), os protótipos são usados principalmente para verificar se os requisitos previamente definidos foram implementados corretamente. No entanto, com orientação adequada dos usuários e análise de seus comentários, essa técnica também pode ser usada para derivar novos requisitos. Pode ser particularmente útil para detectar requisitos não funcionais, insatisfatórios e restrições, ou quaisquer outras características que não possam ser facilmente compreendidas ou definidas antecipadamente em modelos e documentação.

➤ Cenários e storyboards

A palavra *cenário* tem origem no teatro, onde é utilizada para se referir ao esboço de uma peça de teatro, ópera ou similar, indicando uma sequência de cenas com seus personagens. Em TI, usamos esse termo para descrever um fluxo de ações para um sistema, incluindo os usuários envolvidos (a quem costumamos chamar aqui de atores). Por meio de cenários, você pode explorar formas alternativas de realizar um processo em um sistema. Devido à sua estrutura leve, são fáceis de desenvolver e podem ser alterados rapidamente. Da mesma forma que para protótipos, cenários e storyboards podem ser aplicados tanto na elicitação (inicial) quanto na validação (posterior) de requisitos.

Os cenários podem ser documentados de forma escrita ou visual. A forma visual de um cenário é chamado de *Storyboard*.

Um storyboard é tipicamente uma espécie de história em quadrinhos com uma série de painéis que mostram a interação de certas pessoas com o sistema. Consulte a Figura 4.7 para obter um exemplo. Cenários e storyboards são úteis para a elaboração inicial de ideias em termos de processos e atividades.

**Figura 4.7 Exemplo de storyboard**

O *design thinking* não é tanto uma técnica, mas sim um conceito, uma atitude, uma filosofia, uma família de processos e, muitas vezes, uma caixa de ferramentas cheia de técnicas. O foco está na inovação e na resolução de problemas. Existem várias variantes de pensamento de design, principalmente usando técnicas leves, visuais e ágeis. Dois princípios básicos podem ser encontrados em todas as variantes:

Pensamento de design

➤ Empatia

O primeiro passo para os pensadores de design é encontrar o problema real por trás do problema em questão. Eles tentam entender o que as partes interessadas realmente pensam, sentem e fazem quando interagem com um sistema. Portanto, muitas vezes nos referimos ao *design thinking* como *design centrado no ser humano*. Personas, mapeamento de empatia e co-criação com o cliente são técnicas comuns para esse fim.

➤ criatividade

Uma característica comum do *design thinking* é o *diamante*: a alternância de pensamento divergente e convergente. O pensamento divergente visa explorar uma questão de forma mais ampla e profunda, gerando muitas ideias diferentes, e o pensamento convergente foca, seleciona, poda, combina essas ideias em uma única entrega final. Um padrão básico, o modelo *de diamante duplo*, é mostrado na Figura 4.8 (ver [DeCo2007]).

Um tratamento detalhado do pensamento de design está além do escopo deste Manual do Nível Fundamental.

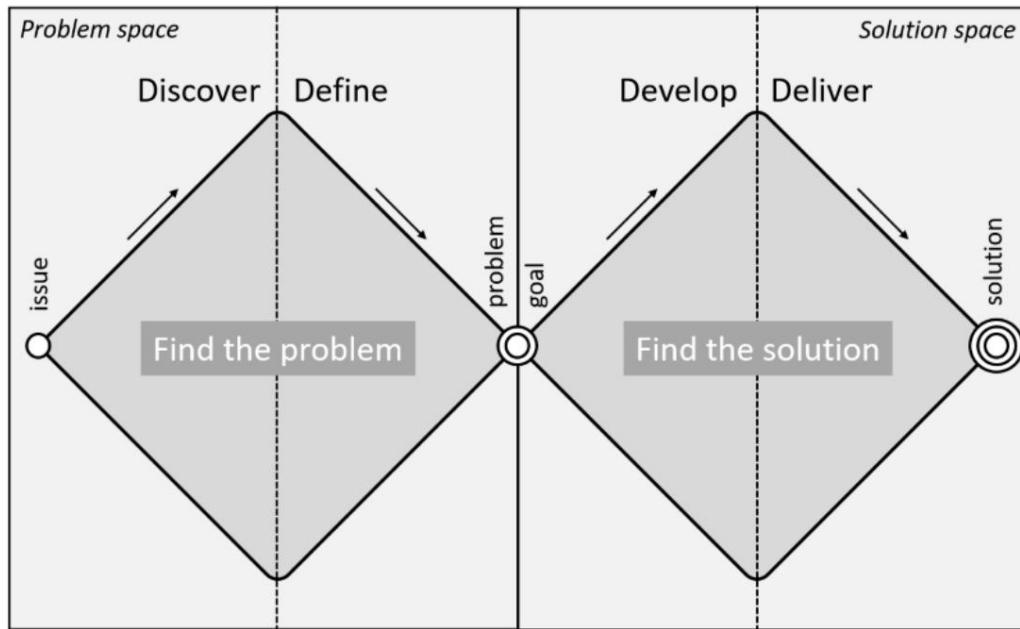


Figura 4.8 O diamante duplo

4.3 Resolvendo Conflitos em Relação aos Requisitos

Durante a elicitação, você reúne uma ampla coleção de requisitos de diferentes fontes, com diferentes técnicas e em diferentes níveis de abstração e detalhe.

Resolução de Conflitos

As técnicas de elicitação que você usa não garantem por si só que esta coleção como um todo forma um conjunto de requisitos único, consistente e acordado que captura a essência do sistema. Durante e após a elicitação de um conjunto de requisitos para um determinado sistema, você pode descobrir que alguns dos requisitos são conflitantes: eles podem ser inconsistentes, incompatíveis, contraditórios. Pode ser que os requisitos entrem em conflito (por exemplo, "todo o texto deve ser preto no branco" versus "todas as mensagens de erro devem ser vermelhas") ou que algumas partes interessadas tenham uma opinião diferente sobre o mesmo requisito (por exemplo, "todas as mensagens de erro deve ser vermelho" versus "as mensagens de erro do usuário devem ser vermelhas, todas as outras mensagens de erro devem ser azuis"). Como não podemos desenvolver um (parte específica de um) sistema com base em requisitos conflitantes, os conflitos devem ser resolvidos antes do início do desenvolvimento. Como Engenheiro de Requisitos, você é quem deve garantir que todas as partes interessadas cheguem a um entendimento compartilhado (consulte o Capítulo 2, Princípio 3) do conjunto completo de requisitos, desde que sejam relevantes para eles e que concordem com esse conjunto .

Mas o que é um conflito? Um conflito é um certo desacordo entre pessoas: "Uma interação entre agentes (indivíduos, grupos, organizações, etc.), onde pelo menos um agente percebe incompatibilidades entre seu pensamento/ideias/percepções e/ou sentimentos e/ou vontade e o de outro agente (ou agentes), e se sente restrinido pela ação do outro" [Glas1999]. Em um conflito de requisitos, dois ou mais stakeholders têm uma opinião diferente ou até mesmo contraditória em relação a um determinado requisito ou seus requisitos não podem ser implementados em um determinado sistema ao mesmo tempo; veja a Figura 4.9.

Conflito

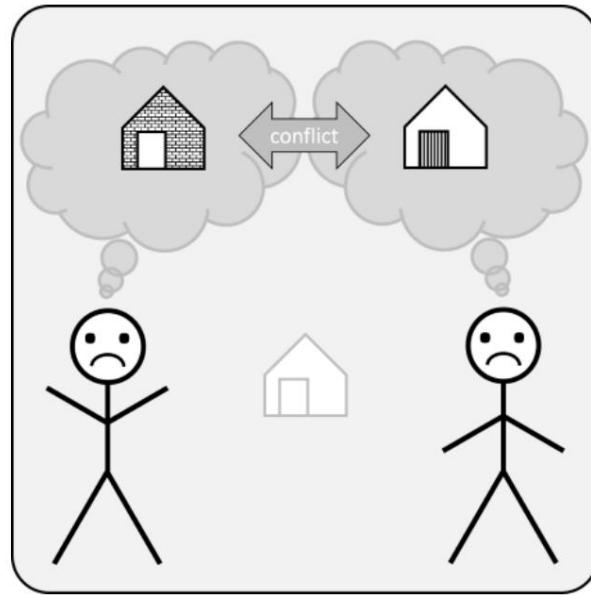


Figura 4.9 Um conflito de requisitos

Lidar com conflitos de requisitos pode ser difícil, doloroso e demorado, especialmente quando estão envolvidos problemas pessoais. No entanto, negar ou ignorar conflitos não é uma opção, portanto o Engenheiro de Requisitos deve procurar ativamente maneiras de resolvê-los. No final, todas as partes interessadas devem entender e concordar com todos os requisitos que são relevantes para elas. Se algumas partes interessadas não concordarem, esta situação deve ser reconhecida como um conflito que deve ser resolvido de acordo.

4.3.1 Como resolver um conflito de requisitos?

Para resolver um conflito de requisitos corretamente, os seguintes passos devem ser seguidos:

Passos para resolver conflitos

► Identificação de conflito

Muitas vezes temos conflitos em nosso dia a dia. Eles nos dão uma sensação desagradável, então uma estratégia comum é simplesmente evitá-los, ignorá-los ou negá-los. Isso pode dificultar a localização de conflitos. A maioria deles tende a ficar escondida e só pode ser detectada por observação cuidadosa. Existem muitos indicadores aos quais você pode prestar atenção, tanto na comunicação quanto na documentação: Na comunicação, você pode observar comportamentos como negação, indiferença, pedantismo, pedir continuamente mais detalhes, interpretações deliberadamente incorretas, ocultação ou delegação.

Na documentação, você pode encontrar coisas como declarações contraditórias das partes interessadas, resultados conflitantes da análise de documentos ou sistemas, inconsistências em diferentes níveis de detalhe e uso inconsistente de termos.

Se você observar tais indicadores, isso não significa necessariamente que haja um conflito de requisitos, mas certamente você deve desconfiar. Uma discussão minuciosa com as partes interessadas pode trazer à tona um conflito oculto.

► Análise de conflito

Uma vez identificado um conflito, o Engenheiro de Requisitos deve primeiro esclarecer se esse conflito é um conflito de requisitos ou não. Afinal, um conflito de requisitos é responsabilidade primária dos Requisitos

Engenheiro; outros conflitos podem ser resolvidos por outros participantes, como um gerente de departamento ou um líder de equipe. O Engenheiro de Requisitos deve entender completamente a natureza do conflito de requisitos antes de tentar resolvê-lo. Isso significa que você terá que coletar mais informações sobre o conflito em si e as partes interessadas envolvidas.

Muitos aspectos merecem atenção:

- o *Assunto*: o escopo, o problema ou a questão real por trás do conflito.
- o *Requisitos afetados*: quais requisitos específicos são afetados? o *Partes interessadas envolvidas*: quem discorda de quem sobre o quê? o *Opiniões* das partes interessadas: deixe-as expor o seu ponto de vista da forma mais clara possível para que todas as partes em conflito entendam os fundamentos emitir.
- o A *causa* do conflito: qual é a razão por trás da diferença de opiniões?
- o A *história* do conflito: o que aconteceu antes que influencia essas opiniões agora?
- o *Consequências*: os custos e riscos estimados associados à resolução ou não resolução do conflito.
- o *Restrições do projeto*: restrições pessoais, organizacionais, específicas de conteúdo ou específicas de domínio podem determinar o espaço da solução.
- o A análise dessas informações ajudará a reconhecer o tipo de conflito (para mais informações, consulte a Seção 4.3.2) e indicará as formas de resolvê-lo.

► Resolução de conflitos

Uma vez alcançado um entendimento profundo da natureza do conflito de requisitos, da atitude das partes interessadas envolvidas e das restrições do projeto, o Engenheiro de Requisitos selecionará uma técnica de resolução adequada. Muitas técnicas podem ser usadas, conforme explicado na Seção 4.3.3. O primeiro passo deve ser sempre fazer com que a técnica escolhida seja aceita pelos stakeholders envolvidos antes de aplicá-la. Se alguns stakeholders não concordarem de antemão com a aplicação de determinada técnica, certamente não aceitarão o resultado dela, portanto, ao final, o conflito não será resolvido. Em princípio, o Engenheiro de Requisitos não é um dos stakeholders envolvidos, então você pode e deve aplicar as técnicas de resolução selecionadas de forma objetiva, estritamente neutra, e acolher qualquer resultado resultante da aplicação da técnica.

► Documentação de resolução de conflitos

A resolução de conflitos pode influenciar os requisitos de uma forma que não seja óbvia para alguém que não esteve envolvido no conflito. O conjunto de requisitos resultante pode parecer ilógico ou ineficiente. Assim, a resolução de conflitos deve ser devidamente documentada e comunicada no que diz respeito a aspectos como os seguintes:

- o Pressupostos sobre o conflito e sua resolução
- o Potenciais alternativas consideradas
- o Restrições que influenciam a técnica e/ou resolução escolhida o A forma como
- o conflito foi resolvido, incluindo as razões da escolha
resolução
- o Tomadores de decisão e outros colaboradores
- o Se você não documentar a resolução, depois de um tempo, as partes interessadas
podem simplesmente esquecer ou ignorar as decisões que foram tomadas. E mais
tarde no projeto, os desenvolvedores podem não entender a lógica por trás de um
design de sistema específico e podem implementá-lo de uma maneira diferente.

Você não precisa ter medo de conflitos de requisitos, pois eles sempre ocorrerão. Isso não deve ser uma surpresa para você; na verdade, você deve se preocupar se não detectar nenhum conflito. Eles são bastante comuns, portanto, se você não os encontrar, provavelmente perdeu alguns. Mas nunca os ignore. Se você não resolver todos os conflitos de requisitos que notar imediatamente, eles aparecerão mais tarde no processo de desenvolvimento. E como Barry Boehm [Boeh1981] já descobriu há muito tempo, quanto mais tarde você descobrir um problema, mais caro será resolvê-lo.

4.3.2 Tipos de Conflitos

Para obter uma melhor compreensão da natureza de um conflito, é útil distinguir entre diferentes tipos de conflito. Isso ajuda na seleção de técnicas de resolução adequadas.

Nós discernimos seis tipos de conflito:

Tipos de conflito

➤ Conflito de assunto

Um conflito *de assunto* ocorre quando as partes conflitantes realmente têm diferentes necessidades factuais, causadas principalmente pelo uso pretendido do sistema em diferentes ambientes. Um bom exemplo é um sistema que será utilizado em diferentes países, cada um com sua própria legislação. Pode ser difícil resolver tal conflito porque os fatos subjacentes não podem ser alterados. A primeira coisa a fazer é analisar e documentar detalhadamente esses fatos e fazer com que as partes conflitantes concordem sobre a natureza exata do conflito.

➤ Conflito de dados

Um *conflito de dados* está presente quando algumas partes se referem a dados inconsistentes de fontes diferentes ou interpretam os mesmos dados de maneira diferente. Isso pode ser devido a má comunicação, dados de fundo ausentes, diferenças culturais, preconceitos existentes, etc. Estimativas em particular, como vendas futuras, podem facilmente gerar um conflito de dados, pois muitas vezes são baseadas em suposições. Detectar um conflito de dados não é fácil, porque como um Engenheiro de Requisitos, você pode pensar que suas próprias fontes estão corretas e sua própria interpretação é auto-evidente.

Devido a esse viés, você costuma suspeitar de outro tipo de conflito no início.

Compreender como as pessoas podem chegar a uma interpretação diferente requer muita empatia. A comunicação - repetidamente - é fundamental para detectar e resolver esse tipo de conflito.

➤ Conflito de interesses

Um *conflito de interesses* é baseado em diferentes posições das partes conflitantes, formadas por objetivos pessoais, objetivos relacionados a um grupo ou objetivos relacionados a um papel. Você deve entender as preocupações e necessidades das partes interessadas envolvidas antes de resolver esse tipo de conflito. No entanto, tenha em mente que, no caso de interesses pessoais, as partes interessadas geralmente não revelam seus verdadeiros motivos e apresentam argumentos aparentemente factuais, mas essencialmente artificiais. Se uma discussão for sobre um conflito de interesses, você pode observar as partes em conflito tentando convencer umas às outras a seguir seus argumentos e entender as necessidades da função ou grupo. A resolução pode se beneficiar da identificação e fortalecimento de interesses compartilhados. Trabalhar em um entendimento mútuo sobre os ganhos e dores de ambas as partes pode ser um ponto de partida para encontrar uma solução.

➤ Conflito de valor

Um *conflito de valores* é baseado em diferenças de valores e princípios das partes interessadas envolvidas. Comparado a um conflito de interesses, um conflito de valores é mais individual e relacionado a perspectivas globais e de longo prazo. Os valores são mais estáveis do que os interesses e raramente mudam no curto prazo. Se um conflito de valores for o motivo de uma discussão, as partes em conflito enfatizarão por que seus argumentos são importantes do seu ponto de vista, revelando seus valores e princípios internos. Eles tendem a insistir em seus argumentos e não estão dispostos a desistir. Para resolver tais conflitos, busque valores superiores que unam as partes. Os conflitos de valores são notoriamente difíceis de resolver e alcançar a compreensão mútua e o reconhecimento dos princípios de cada um é o melhor que você pode obter.

➤ conflito de relacionamento

Um *conflito de relacionamento* geralmente é baseado em experiências negativas com outra parte no passado ou em situações comparáveis com pessoas semelhantes. Muitas vezes, emoções e falta de comunicação estão envolvidas, o que torna o conflito muito mais difícil de resolver. As partes em conflito abusam das discussões sobre os requisitos para expressar sua raiva pelo comportamento umas das outras, esquecendo-se de fatos, números e justiça. Trazer a discussão de volta aos requisitos raramente ajudará; às vezes, unir as partes em torno de um valor mais alto é bem-sucedido. Na maioria dos casos, você terá que encaminhar o problema para outras partes interessadas ou para um nível superior de autoridade; trocar pessoas é uma resolução potencial. Esteja ciente de que um conflito de relacionamento geralmente ocorre simultaneamente com outros tipos de conflito - por exemplo, um conflito de interesses. Analisar a causa raiz e resolver o outro tipo de conflito pode ser a melhor maneira de melhorar o relacionamento.

➤ conflito estrutural

Chamamos um conflito *de estrutura* quando envolve desigualdade de poder, competição por recursos limitados ou dependências estruturais entre as partes. O desequilíbrio resultante (muitas vezes percebido apenas por uma das partes) causa problemas na comunicação e na tomada de decisões. Outra razão para tais conflitos pode ser restrições de recursos ou dependências de produtos de trabalho a serem entregues por outra parte. As partes podem usar a discussão sobre os requisitos para alterar ou preservar o status quo. A hierarquia pode ser

mal utilizado para forçar as decisões. Também para conflitos estruturais, muitas vezes é necessário escalar o problema para outras partes interessadas ou para um nível mais alto de autoridade.

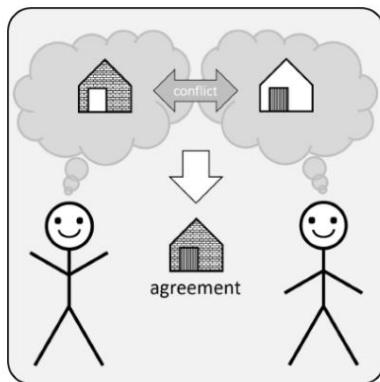
A maioria dos conflitos de requisitos pode ser categorizada como conflito de assunto, dados, interesse ou valor. Os conflitos estruturais e de relacionamento geralmente não estão diretamente relacionados aos requisitos e, portanto, o Engenheiro de Requisitos pode não ser a parte apropriada para resolvê-los. No entanto, na realidade, a maioria dos conflitos se enquadra em mais de uma categoria, pois diferentes causas interagem. Portanto, é aconselhável estar atento a todos os tipos de conflitos, mesmo que a solução não seja de sua responsabilidade. Se outra pessoa resolver o conflito, certifique-se de que isso aconteça; enquanto um conflito não for resolvido, ele continuará a ter um impacto negativo em seu trabalho como Engenheiro de Requisitos.

Conflitos mistos

4.3.3 Técnicas de Resolução de Conflitos

Dependendo do tipo e do contexto (stakeholders, restrições, etc.) de um conflito, uma técnica de resolução adequada é selecionada. Técnicas comumente usadas incluem [PoRu2015]:

Acordo



Acordo

Um acordo resulta de uma discussão entre as partes interessadas envolvidas, a ser continuada até que eles entendam completamente as posições de cada um e concordem com uma determinada opção preferida por todas as partes. Pode ser muito demorado, especialmente quando várias partes estão envolvidas. Se for bem-sucedido, fornecerá motivação adicional às partes interessadas, de modo que o resultado tem boas chances de ser duradouro. Esforçar-se para chegar a um acordo é comum em

conflitos de dados. Se esta técnica não for bem-sucedida dentro de um prazo aceitável, outras técnicas podem ser usadas posteriormente.

Compromisso

Compromisso

Um compromisso é bastante semelhante a um acordo. Aqui, no entanto, as partes interessadas concordam com uma opção que não é de sua preferência, mas com a qual podem conviver, porque aceitar o compromisso é considerado melhor do que continuar o conflito. Portanto, um compromisso também pode ser duradouro. O compromisso pode conter novos elementos que não estavam presentes nas preferências originais das partes interessadas e que podem ter sido introduzidos pelo Engenheiro de Requisitos. Um bom compromisso é uma alternativa em que todas as partes se sentem confortáveis como o equilíbrio de abrir mão de coisas e receber algo em troca. Um compromisso geralmente é o próximo na fila se um acordo não puder ser alcançado a tempo. É adequado para conflitos de assunto e também pode funcionar para conflitos estruturais e de interesse.



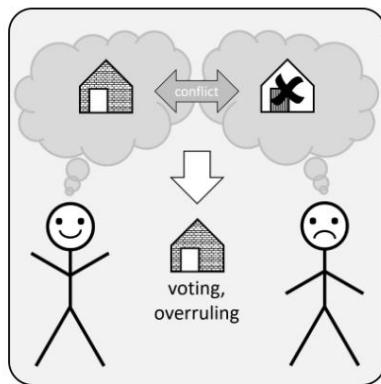
Votação

Votação

A votação funciona melhor quando uma escolha relativamente simples deve ser feita entre um conjunto claro de requisitos conflitantes. As partes interessadas que participam da votação (geralmente não apenas as partes conflitantes, mas todas as partes interessadas envolvidas) devem entender completamente as alternativas e as consequências de seu voto.

Para evitar influências de dependências ou desequilíbrio de poder, a votação é melhor feita

anonimamente e com um moderador neutro. O procedimento de votação em si deve ser acordado entre as partes interessadas antes da votação real. A votação é um meio rápido e fácil de resolução de conflitos, mas a parte que perder a votação ficará desapontada e pode precisar de atenção.



A votação pode funcionar para a maioria dos tipos de conflito e pode ser uma boa maneira de resolver conflitos de interesse e assuntos.

Overruling

Overruling

Se um acordo ou compromisso não puder ser alcançado e pelo menos uma das partes conflitantes se recusar a participar da votação, *overruling* pode ser uma opção. Muitas vezes é aplicado sob pressão, quando não há tempo suficiente para usar técnicas mais convenientes. Normalmente, o *overruling* é feito transferindo a escolha entre os requisitos conflitantes para um tomador de decisão que é mais alto em autoridade ou hierarquia do que todas as partes conflitantes e tem poder suficiente para que a decisão seja implementada. Portanto, é uma boa forma de resolver conflitos de interesse e estruturais. Nessa situação, é particularmente importante que o tomador de decisão compreenda plenamente as alternativas, a posição das partes conflitantes e as consequências da decisão. Uma variante do *overruling* é terceirizar a decisão para terceiros — por exemplo, um especialista externo. Nesse caso, é importante primeiro obter um acordo entre as partes interessadas sobre o tomador de decisão. Assim como na votação, você pode precisar prestar atenção ao *perdedor*.

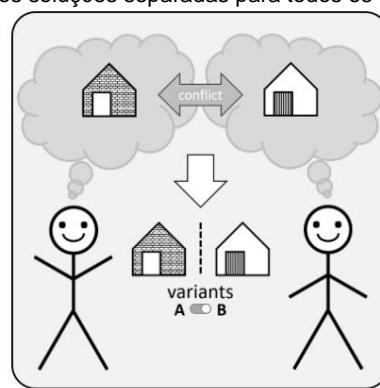
Definição de variantes

Definição de variantes

A definição de variantes geralmente é considerada para assuntos, interesses e conflitos de valor. Vimos que não podemos implementar requisitos conflitantes em um único e mesmo sistema. A definição de variantes significa que construímos soluções separadas para todos os requisitos conflitantes.

Isso geralmente é implementado desenvolvendo um sistema que pode ser configurado por meio de parâmetros para exibir os recursos desejados.

Pode parecer uma solução perfeita, mas tem um preço: leva muito tempo para definir a solução e uma complexidade crescente (além de custos adicionais) é introduzida no sistema, tanto para desenvolvimento quanto durante operações e manutenção. Esta técnica é, portanto, viável apenas se tempo e orçamento suficientes estiverem disponíveis.



Técnicas auxiliares

Técnicas auxiliares

Além disso, existem diversas *técnicas auxiliares* que normalmente não são utilizadas isoladamente, mas sim para auxiliar as técnicas citadas acima.

Em *Considerar todos os fatos (CAF)*, você considera soluções alternativas para vários critérios predefinidos, por exemplo, custo, tempo, risco, recursos disponíveis.

CAF

A ponderação desses critérios pode fornecer mais clareza sobre os prós e contras das alternativas e ajudar a identificar a *melhor* alternativa.

Mais-Menos-Interessante (PMI, consulte [DeBo2005]) é uma ferramenta de brainstorming e tomada de decisão. Incentiva o exame de ideias e conceitos de mais de uma perspectiva e, portanto, é valioso para a resolução de conflitos. No PMI, os participantes (geralmente todas as partes interessadas envolvidas) primeiro identificam todos os aspectos positivos (mais) das alternativas, depois os negativos (menos) e, finalmente, os pontos interessantes, coisas que precisam ser mais investigadas. A alternativa com mais vantagens e menos desvantagens é a alternativa preferida.

PMI

Na verdade, tanto o CAF quanto o PMI são variantes da *matriz de decisão*, uma abordagem *metódica* para resolução de conflitos. Os requisitos conflitantes são avaliados com base em um (maior) número de critérios, após o que as pontuações nesses aspectos são usadas para calcular uma pontuação final (ponderada) para as alternativas. A pontuação *mais alta* vence, como a *Alternativa 1* no exemplo da Tabela 4.2 abaixo. Na verdade, a priorização (consulte a Seção 6.8) é então usada como uma técnica de resolução. Como dito anteriormente, essas técnicas geralmente são vistas como auxiliares: elas criam mais informações sobre as alternativas e, assim, ajudam na técnica de resolução escolhida. Eles podem até ser usados como uma única técnica se todas as partes interessadas envolvidas concordarem em aceitar o resultado.

Matriz de decisão

Tabela 4.2 Exemplo de uma matriz de decisão

Critério	Peso	Alternativa 1: somente iOS		Alternativa 2: Android e iOS	
		Pontuação	Pesada	Pontuação	Pesada
Cliente base	2	3	6	4	8
Dev. custo	1	3	3	2	2
mercado tt	3	4	12	2	6
Reputação	2	2	4	4	8
Exp. do usuário	1	5	5	3	3
Total			30		27

4.4 Validação de Requisitos

No Capítulo 2, Princípio 6, enfatizamos a importância de validar os requisitos para evitar partes interessadas insatisfeitas. Como os requisitos formam a entrada para o desenvolvimento subsequente do sistema, devemos garantir sua qualidade desde o início para

*Validação de
requisitos*

reduzir o esforço desperdiçado a jusante, tanto no nível dos requisitos individuais quanto dos produtos de trabalho que os contêm (Figura 4.10).

Devemos validar a cobertura das necessidades das partes interessadas por nossa documentação, o grau de concordância entre todas as partes interessadas e a probabilidade de nossas suposições sobre o contexto do sistema antes de entregarmos os requisitos aos desenvolvedores ou fornecedores. Embora o nível de detalhe possa variar, isso se aplica tanto para abordagens de desenvolvimento iterativas quanto para sequenciais.

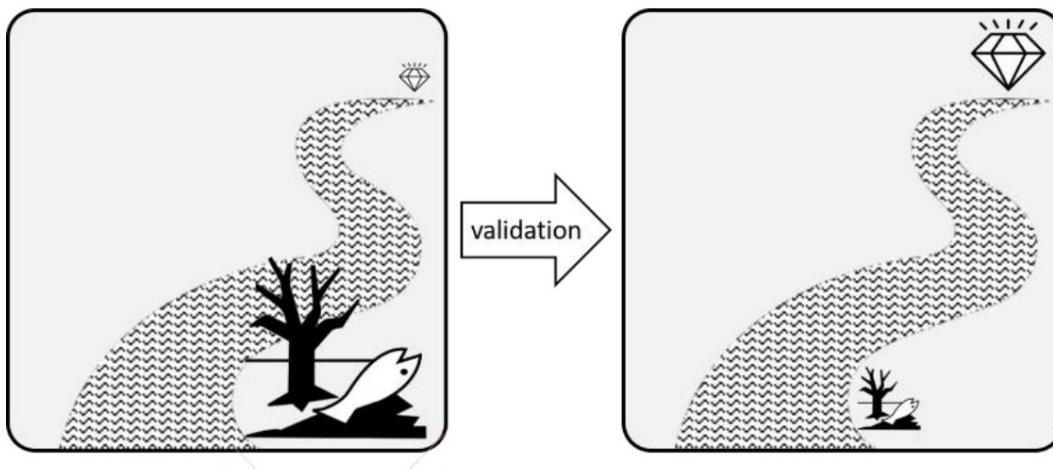


Figura 4.10 A qualidade a montante reduz o desperdício a jusante

A validação agrupa tempo e custo ao projeto, portanto sua eficiência e eficácia devem ser uma preocupação do Engenheiro de Requisitos. Portanto, é importante monitorar e analisar continuamente os defeitos que ocorrem durante o desenvolvimento e na operação. Se a causa raiz de tais defeitos parece estar nos requisitos, o processo de validação de requisitos falhou de alguma forma. Portanto, como Engenheiro de Requisitos, você deve procurar continuamente eativamente por oportunidades para melhorá-lo.

*Melhoria
continua*

4.4.1 Aspectos Importantes para Validação

Em relação ao conceito de validação, alguns aspectos são importantes para obter o máximo valor dele (ver também [PoRu2015]):

- Envolvendo as partes interessadas

Aspectos para validação

corretas Como Engenheiro de Requisitos, você precisa decidir quem deseja convidar para participar da validação. Nesse sentido, um aspecto importante que você deve considerar é o grau de independência entre as pessoas envolvidas na elicitação dos requisitos e as que os validam. Um baixo nível de independência (convidar as partes interessadas que já participaram da elicitação) é barato e fácil de organizar, mas pode ignorar certos defeitos por causa do próprio foco, pontos cegos, interesses conflitantes ou suposições falhas dessas pessoas. Um maior grau de independência (por exemplo, convidando revisores ou auditores externos) leva mais tempo e esforço para organizar e executar e traz custos (iniciais) mais altos, mas pode, a longo prazo, ser mais eficaz para encontrar defeitos mais e mais graves. Consequentemente, maior risco no escopo do projeto e/ou no contexto do sistema exige um maior grau de independência.

➤ Separando a identificação e a correção de defeitos

Pode ser tentador corrigir todos os defeitos assim que forem detectados.

No entanto, isso geralmente não é uma maneira eficiente nem eficaz de trabalhar, pois os defeitos podem influenciar uns aos outros. Um defeito encontrado posteriormente durante a validação pode invalidar a correção de um anterior. Um requisito inicialmente marcado como defeituoso pode se mostrar correto quando todos os requisitos forem estudados. Você pode decidir não corrigir alguns defeitos (menores) em vista do esforço envolvido em relação ao conjunto total de defeitos encontrados. E, afinal, as pessoas envolvidas na validação de requisitos devem se concentrar em encontrar defeitos e não em desenvolver ideias de como corrigi-los. Portanto, a recomendação é selecionar primeiro (um conjunto coerente de) requisitos para validação e decidir se corrigirá ou não certos defeitos encontrados somente após a verificação de todo o conjunto.

➤ Validação de diferentes pontos de vista

Uma validação adequada é sempre um esforço de grupo, não uma atividade realizada pelos Engenheiros de Requisitos por conta própria. Os melhores resultados são alcançados quando a validação é realizada por uma equipe interdisciplinar na qual participantes selecionados contribuem com seus próprios conhecimentos. Em geral, podemos dizer que a entrada, a saída e os pares devem ser representados. Em projetos iterativos, a equipe ágil atual é uma escolha razoável, mas o grau de independência pode ser baixo e validadores adicionais devem ser convidados; em projetos sequenciais, uma equipe específica pode ser composta para cada esforço de validação separado. Dependendo da fase do projeto, informações de negócios, usuários, desenvolvedores, testadores, operadores e gerentes de aplicativos são úteis; às vezes, especialistas no assunto ou especialistas em tópicos como desempenho, segurança e usabilidade podem ser adicionados.

➤ Validação repetida

Em projetos sequenciais, a maioria dos requisitos é elicitada e documentada na fase inicial e validada completamente no final dessa fase. No entanto, este não deve ser o único momento de validação. Durante o restante do projeto, novos insights podem levar à atualização, detalhamento e expansão do conjunto original de requisitos. Isso pode ameaçar a qualidade, coerência e consistência dos requisitos e, portanto, validações adicionais podem ser necessárias. Estes são frequentemente planejados em marcos do projeto.

Em projetos iterativos, muitos dos rituais ágeis incluem esforços de validação. O planejamento do sprint, o refinamento do backlog, as revisões do sprint e até mesmo as reuniões diárias oferecem oportunidades para validar e melhorar os requisitos. No entanto, esses esforços geralmente se concentram em requisitos individuais e detalhados e o quadro geral pode ser negligenciado. Uma validação inicial do backlog completo do produto no início de um projeto ou incremento é um bom começo. Outras iniciativas úteis são sprints de proteção repetidos e validação geral adicional em tempos de lançamento.

4.4.2 Técnicas de Validação

Quanto a outras técnicas, o Engenheiro de Requisitos pode escolher entre uma grande caixa de ferramentas de técnicas de validação que diferem em formalidade e esforço. Muitos fatores influenciam a seleção dessas técnicas - por exemplo, o ciclo de vida de desenvolvimento de software

[Técnicas de validação](#)

modelo, maturidade do processo de desenvolvimento, complexidade e nível de risco do sistema, requisitos legais ou regulatórios e necessidade de trilha de auditoria.

Muitas vezes, no decorrer de um projeto, o grau de esforço e formalidade aumenta no final, pois decisões finais sobre o sistema e sua implementação devem ser tomadas. Além disso, você verá que a quantidade, o valor e o nível de detalhamento do feedback das partes interessadas aumentam à medida que os produtos de trabalho a serem validados se tornam mais concretos e detalhados. Isso implica a aplicação de diferentes técnicas de validação em diferentes etapas do projeto. No início de um projeto, ciclos de validação e feedback curtos e leves são preferidos, como é comum em abordagens ágeis. Isso garante qualidade desde o início. Mais tarde no projeto, prevalecerão técnicas pontuais mais formais e demoradas.

Além disso, também é possível observar uma mudança no foco das atividades de validação. Nas fases iniciais de um projeto, as técnicas são mais utilizadas para validar a especificação de requisitos. Em fases posteriores, o foco das mesmas técnicas pode mudar para a validação de sua *implementação*.

Em geral, discernimos três categorias de técnicas de validação (ver Figura 4.11):

- técnicas de revisão
- técnicas exploratórias
- Desenvolvimento de amostras

Técnicas de revisão e desenvolvimento de amostras são chamadas de estáticas, pois se concentram em analisar as especificações de um sistema sem executá-lo. Nas técnicas exploratórias, a validação foca no comportamento real (ou simulado) do sistema em operação; essas técnicas são chamadas dinâmicas.

Estático vs dinâmico

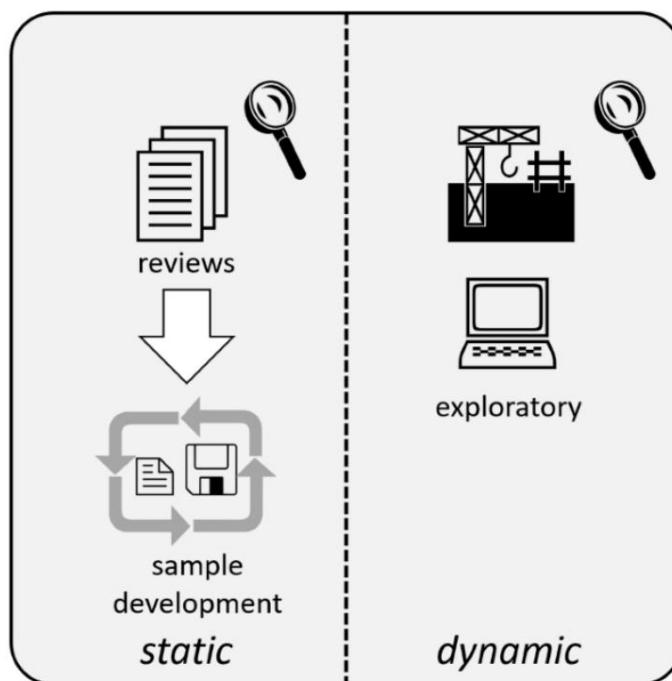


Figura 4.11 Categorias de técnicas de validação

A característica comum das *técnicas de revisão* é que elas dependem do estudo visual dos produtos de trabalho iniciais e intermediários. Eles variam de informais a muito formais e podem ser aplicados desde o início de um projeto até a implementação do sistema.

técnicas de revisão

Na maioria dos casos, a revisão dos requisitos é limitada às fases anteriores de um projeto.

Normalmente, em uma revisão, verificamos produtos de trabalho *estáticos* que definem ou descrevem como o sistema deve funcionar. Para obter mais informações sobre revisão, consulte [OleA2018].

As *revisões informais* geralmente seguem o ciclo autor-revisor. Um autor envia um produto de trabalho para um grupo de pessoas com a solicitação de validá-lo. Normalmente, este é um pequeno grupo de membros da equipe, colegas e/ou usuários envolvidos no projeto. Os autores podem selecionar o grupo por conta própria ou sua composição pode ser prescrita por regulamentos da empresa. Após um período curto (mas geralmente não predefinido), o autor coleta todos os comentários da revisão e os utiliza para atualizar o produto de trabalho em questão. É uma boa prática documentar os comentários em um registro de revisão e acompanhar a maneira como eles são processados. No entanto, devido à natureza informal deste tipo de revisão, os autores são livres para decidir se e como usar os comentários. Frequentemente, a revisão é repetida em vários rascunhos até que o autor esteja satisfeito com a qualidade.

avaliações informais

Como são informais, você pode esperar pouco benefício desses tipos de revisões para validar e melhorar a qualidade dos requisitos. No entanto, se todos os participantes estiverem comprometidos com a qualidade e puderem e desejarem dedicar tempo suficiente ao processo de revisão, as revisões informais são um meio de validação fácil, barato e acessível.

Na verdade, essa abordagem é comum para rascunhos iniciais. Para a versão final de um produto de trabalho, uma técnica mais formal pode ser uma escolha melhor.

As *revisões formais* seguem uma forma prescrita de trabalho. Eles são frequentemente usados para produtos de trabalho importantes ou marcos, para versões finais e em situações em que riscos elevados estão em jogo. Embora existam muitos tipos de revisões formais, elas podem ser divididas em dois grupos principais:

Revisões formais

Walkthroughs A

A essência de um *walkthrough* é que o autor de um produto de trabalho o explica passo a passo para um público em uma sessão interativa. Na prática, os walkthroughs vêm em duas variantes, onde (1) os revisores entram na reunião sem nenhuma preparação e ouvem o autor, fazendo perguntas ad hoc; ou (2) obtêm o produto do trabalho antes da reunião e preparam perguntas para o autor. Os participantes da audiência podem fazer comentários, identificar falhas e sugerir alternativas. O autor dá mais explicações, se necessário, e pode discutir soluções para os pontos fracos identificados e ponderar alternativas em relação às ideias originais. Há duas ocasiões em que os walkthroughs são mais bem aplicados: (a) em uma fase inicial do projeto para discutir a viabilidade de um determinado conceito de sistema ou esboço de solução; e (b) na transferência de um produto de trabalho intermediário para outra parte que o utilizará como insumo para desenvolvimento subsequente.

Em projetos iterativos, os walkthroughs estão presentes principalmente na forma de sessões regulares de refinamento antes de uma iteração e revisões de sprint no final dela.

Inspeções

As *inspeções* estão entre as técnicas de revisão mais formais. Aqui, a responsabilidade pela revisão não é do autor, mas de um líder de revisão independente, muitas vezes chamado de *moderador*. Uma inspeção é normalmente realizada na forma de uma reunião com o moderador, o autor e um grupo de *inspetores*.

Os inspetores são selecionados entre colegas, empresas, usuários e/ou especialistas.

Eles são solicitados a verificar o produto de trabalho com base em seus conhecimentos específicos, para verificar sua aderência aos padrões, normas e regulamentos aplicáveis e para avaliá-lo em relação aos objetivos acordados. Freqüentemente, essa verificação pelos inspetores é realizada durante uma preparação individual minuciosa antes da reunião real, guiada por listas de verificação detalhadas. Na reunião de revisão, o autor participa como ouvinte, explicando coisas que não estão claras e tentando entender os comentários dos fiscais e as consequências para o produto do trabalho.

Normalmente, uma inspeção segue um processo rigoroso e documentado que é gerenciado pelo moderador e se concentra em encontrar defeitos e medir aspectos de qualidade definidos e fornecer uma trilha de auditoria detalhada. Dessa forma, as inspeções são frequentemente usadas para decidir sobre a liberação de um produto de trabalho para uma próxima etapa do processo de desenvolvimento ou até mesmo para a implementação final. As inspeções são aplicadas principalmente em sistemas críticos (de segurança) e processos de negócios. Nas abordagens ágeis, essa forma formal de revisão é incorporada à própria metodologia – por exemplo, com as cerimônias do Scrum (refinamento, planejamento, revisão do sprint).

As *técnicas exploratórias* oferecem a um grupo de partes interessadas e possíveis usuários a oportunidade de obter experiência prática com uma versão intermediária (parte do) do sistema em desenvolvimento. Em contraste com as revisões, as técnicas exploratórias são *dinâmicas*: elas observam o comportamento (real ou simulado) do sistema operacional conforme experimentado pelos usuários por meio das interfaces de usuário. Os participantes são convidados a utilizar o sistema de forma semelhante ao uso pretendido na produção. Eles são relativamente livres para fazê-lo, mas às vezes certas orientações são dadas. Após um período de uso, os participantes relatam suas experiências e seus comentários sobre o comportamento atual do sistema ao Engenheiro de Requisitos. Isso pode incluir defeitos encontrados e sugestões de melhoria.

técnicas exploratórias

Técnicas exploratórias são comuns em abordagens de desenvolvimento iterativas e de design thinking. De fato, o desenvolvimento incremental usual, começando com o lançamento de um *produto mínimo viável (MVP)*, seguido pela adição de mais funcionalidades passo a passo, enquanto se mede cuidadosamente as reações do mercado e se ajusta o sistema de acordo, pode ser visto como uma validação exploratória dos requisitos na produção.

Desenvolvimento iterativo

Técnicas exploratórias comuns incluem:

► Prototipagem

Na validação com *prototipagem*, uma versão inicial específica do sistema é fornecida a um grupo de interessados para avaliação. Esta versão pode ser construída explicitamente para fins de validação, após o que é descartada; chamamos isso de protótipo exploratório ou descartável. Claro, protótipos evolutivos, que são continuamente atualizados e estendidos até chegarem ao produto final, também podem ser usados para validação durante seu desenvolvimento. A essência de qualquer protótipo é que, do lado de fora, ele se pareça com o sistema pretendido, permitindo que as partes interessadas ganhem experiência prática enquanto a estrutura interna ainda pode estar inacabada, inoperante ou até mesmo ausente. Ao usar um protótipo para validação, você pode construí-lo para verificar uma característica específica, como interface do usuário, segurança ou desempenho.

➤ Elicitação e validação andam juntas

Como vimos na Seção 4.2.3, a prototipagem e o storyboard também podem ser usados como técnicas de elicitação. Na verdade, essas técnicas suportam tanto a elicitação quanto a validação, andando de mãos dadas: ao validar os requisitos eliciados em um momento anterior, é quase certo que você detectará novos requisitos no feedback dos participantes. Ambos os aspectos da prototipagem são muito proeminentes nas abordagens de design thinking (consulte [LIOg2011]).

➤ Teste alfa e teste beta

Nos testes alfa e beta, uma versão de pré-produção com todos os recursos e totalmente funcional do sistema é fornecida aos usuários finais para operação com os processos de negócios pretendidos em um ambiente realista.

O teste alfa é feito no site do desenvolvedor em um ambiente simulado. O grupo de participantes é relativamente pequeno, algumas orientações podem ser dadas e é possível observar a interação dos usuários com o sistema – por exemplo, em um laboratório de usabilidade.

teste alfa

O teste beta é realizado nos sites do usuário final em produção real (ou em qualquer ambiente que os usuários finais decidam). O sistema é oferecido (principalmente de graça) a um grupo de usuários (às vezes selecionado, mas geralmente desconhecido), com a solicitação implícita de validar sua aparência e comportamento. No teste beta, é importante estimular todos os participantes a darem seu feedback e fornecer uma maneira fácil de fazê-lo. A análise desse feedback após um período prolongado de uso pode fornecer pistas valiosas sobre a qualidade dos requisitos. É particularmente útil para verificar certas suposições feitas durante a elicitação e o desenvolvimento.

Teste beta

➤ teste A/B

O teste A/B geralmente é realizado com uma versão lançada do sistema no ambiente totalmente operacional, mas também pode ser aplicado com versões de pré-lançamento em um ambiente de teste protegido. A essência do teste A/B é que o sistema é oferecido a diferentes grupos de usuários (a maioria selecionados aleatoriamente) em duas variantes que diferem em design ou funcionalidade e realizam os objetivos do usuário de maneira diferente. A reação de ambos os grupos é medida e comparada; isso funciona melhor quando os grupos são grandes o suficiente para permitir a análise estatística. A análise fornecerá informações sobre a qualidade dos requisitos subjacentes e sobre a correção das suposições anteriores.

O teste A/B tem um papel proeminente no *The Lean Startup*, uma das abordagens de design thinking (ver [Ries2011]).

No desenvolvimento de amostra, você fornece um conjunto de requisitos como entrada para desenvolvedores; eles tentam produzir alguns produtos de trabalho intermediários comuns (por exemplo, projetos, código, casos de teste, manuais) com base nessa entrada. O sistema em si não está operacional (ainda), então esse tipo de validação é *estático*, assim como na revisão. Durante esse esforço, os desenvolvedores podem detectar falhas como falta de clareza, omissões e inconsistências que os impedem de produzir a saída pretendida. Claro, essas falhas serão corrigidas. Ao mesmo tempo, porém, a quantidade e gravidade das falhas detectadas é um indicativo da qualidade dos requisitos. Se essa qualidade não for suficiente, é necessária mais validação – por exemplo, revisões adicionais.

Desenvolvimento de amostras

Uma validação semelhante pode ser realizada pelos próprios Engenheiros de Requisitos. Nesse caso, você tenta documentar um conjunto de requisitos em uma forma de representação diferente do tipo original: comumente, convertendo uma especificação de requisitos criada em linguagem natural em um modelo relevante, ou um modelo específico em uma descrição textual. Este exercício é especialmente útil para detectar omissões. Se você encontrar problemas sérios nessa conversão, isso indica a necessidade de validação adicional.

Convertendo tipos de documentação

4.5 Leitura Adicional

Glinz e Wieringa [GIWi2007] explicam a noção e a importância dos stakeholders. Alexander [Alex2005] discute como classificar as partes interessadas. Bourne [Bour2009] lida com a gestão das partes interessadas. Lim, Quercia e Finkelstein [LiQF2010] investigam o uso de redes sociais para análise de stakeholders. Humphrey [Hump2017] discute as pessoas dos usuários.

Zowghi e Coulin [ZoCo2005] apresentam uma visão geral das técnicas de elicitação de requisitos. Gottesdiener [Gott2002] escreveu um livro clássico sobre workshops em ER. Carrizo, Dieste e Juristo [CaDJ2014] investigam a seleção de técnicas de elicitação adequadas.

Maalej, Nayebi, Johann e Ruhe [MNJR2016] discutem o uso de feedback explícito e implícito do usuário para eliciar requisitos. Maiden, Gitzikis e Robertson [MaGR2004] discutem como a criatividade pode fomentar a inovação em ER.

O livro de Moore [Moor2014] é um clássico sobre gestão de conflitos. Glasl [Glas1999] discute como lidar com conflitos. Grünbacher e Seyff [GrSe2005] discutem como chegar a um acordo negociando sobre requisitos ao validar requisitos ou resolver conflitos.

A validação é abordada em qualquer livro de RE; ver [Pohl2010], por exemplo.

5 Processo e Estrutura de Trabalho

Sempre que o trabalho deve ser feito de forma sistemática, um *processo* é necessário para moldar e estruturar a forma de trabalhar e a criação de produtos de trabalho.

Processo

Definição 5.1. Processo: Um conjunto de atividades inter-relacionadas executadas em uma determinada ordem para processar informações ou materiais.

Um processo de Engenharia de Requisitos (RE) organiza como as tarefas de RE são executadas usando práticas apropriadas e produzindo produtos de trabalho necessários. No entanto, não existe um processo de RE comprovado e único para todos (consulte a Seção 1.4). Conseqüentemente, os Engenheiros de Requisitos precisam configurar um processo de RE personalizado que se ajuste à situação em questão.

Necessidade de um processo de RE personalizado

O processo de RE molda o fluxo de informações e o modelo de comunicação entre os participantes envolvidos em RE (por exemplo, clientes, usuários, engenheiros de requisitos, desenvolvedores e testadores). Também define os produtos de trabalho de ER a serem usados ou produzidos. Um processo de ER adequado fornece a estrutura na qual os Engenheiros de Requisitos extraem, documentam, validam e gerenciam os requisitos.

Molda o fluxo de informações e a comunicação

Neste capítulo, você aprenderá sobre os fatores que influenciam o processo de ER e como configurar um processo adequado a partir de um conjunto de facetas de processo.

5.1 Fatores de Influência

Há uma variedade de fatores de influência a serem considerados ao configurar um processo de RE.

Antes de iniciar a configuração de um processo de RE, esses fatores precisam ser investigados e analisados.

Fatores influentes importantes

Por um lado, tal análise fornece informações sobre como configurar o processo de ER. Por exemplo, quando a análise indica que as partes interessadas têm apenas uma vaga ideia sobre seus requisitos, deve ser escolhido um processo de ER que apoie a exploração dos requisitos. Por outro lado, os fatores influenciadores restringem o espaço de possíveis configurações do processo. Por exemplo, se as partes interessadas estiverem disponíveis apenas no início de um projeto de desenvolvimento de sistema, um processo baseado no feedback contínuo das partes interessadas não seria adequado. Abaixo, discutimos fatores importantes para o processo de RE.

Ajuste geral do processo. Ao definir ou configurar um processo de RE, é vital conhecer e entender o processo de desenvolvimento geral escolhido para o sistema a ser desenvolvido — definir um processo de RE que não se encaixa no processo geral não faz sentido. O processo geral pode exigir produtos de trabalho que o processo de RE deve entregar. A terminologia usada para o processo de ER deve estar alinhada com a terminologia do processo geral. Em particular, a terminologia para os produtos de trabalho deve ser alinhada.

Ajuste geral do processo

Isso ajuda a evitar confusões e mal-entendidos. Também facilita a introdução do processo de RE, bem como o treinamento e coaching das pessoas que devem trabalhar de acordo com o processo. Por exemplo, se o sistema for desenvolvido usando um processo linear orientado a planos que depende da existência de uma especificação de requisitos de sistema abrangente e um glossário de sistema no final da fase de requisitos, o processo de ER escolhido deve se encaixar na fase de requisitos de o processo geral e produzir os dois produtos de trabalho necessários.

Contexto de desenvolvimento. O contexto de desenvolvimento também informa o processo de ER. Coisas a considerar incluem o relacionamento cliente-fornecedor-usuário, tipo de desenvolvimento,

Contexto de desenvolvimento

questões contratuais e confiança. Ao analisar o contexto de desenvolvimento, algumas perguntas precisam ser respondidas:

- Relacionamento cliente-fornecedor-usuário: Existe um cliente designado que solicita o sistema e paga por ele e um fornecedor que desenvolve o sistema?
Cliente e fornecedor fazem parte da mesma organização ou pertencem a organizações diferentes? Se for o primeiro caso, quais pessoas atuam no papel de cliente e quais atuam como fornecedor? Quem são os usuários do sistema?
Os usuários pertencem à organização do cliente? Se não, eles usam o sistema como um produto ou serviço para interagir com o cliente (por exemplo, em negócios eletrônicos) ou compram o sistema como um produto ou serviço do cliente (por exemplo, um aplicativo móvel)?
- Tipo de desenvolvimento: Qual é a estrutura organizacional para o desenvolvimento de um sistema? Os tipos típicos incluem:
 - o Um fornecedor especifica e desenvolve um sistema para um cliente específico quem vai usar o sistema.
 - o Uma organização desenvolve um sistema com a intenção de vendê-lo como um produto ou serviço para muitos clientes em um determinado segmento de mercado.
 - o Um fornecedor configura um sistema para um cliente a partir de um conjunto de componentes prontos.
 - o Um fornecedor aprimora e desenvolve um produto existente.
- Contrato: Existe um contrato ou acordo similar que defina formalmente entregáveis, custos, prazos, responsabilidades, etc.? Os contratos podem ser contratos clássicos de preço fixo entre um cliente e um fornecedor, com funcionalidade, prazos e custos fixos, ou podem apenas fornecer uma estrutura financeira, enquanto a funcionalidade é definida iterativamente.
- Confiança: As partes envolvidas confiam umas nas outras? Se, por exemplo, o cliente e o fornecedor não confiam um no outro, os requisitos devem ser especificados com mais detalhes do que seria necessário em um relacionamento baseado em confiança.

Disponibilidade e capacidade das partes interessadas. A disponibilidade das partes interessadas restringe as opções de configuração para o processo de RE. Por exemplo, um processo que requer uma interação próxima e contínua com as partes interessadas não pode ser escolhido se as principais partes interessadas estiverem disponíveis apenas por um curto período de tempo no início do processo.

Disponibilidade e capacidade das partes interessadas

A capacidade das partes interessadas também influencia o processo: quanto menos as partes interessadas forem capazes de expressar suas necessidades com clareza e menos conhecem suas necessidades reais, mais o processo de ER deve acomodar a exploração dos requisitos.

Compreensão compartilhada

Compreensão compartilhada. Apenas uma pequena Engenharia de Requisitos é necessária quando há um alto grau de entendimento compartilhado (consulte o Capítulo 2, Princípio 3) entre as partes interessadas, Engenheiros de Requisitos, designers e desenvolvedores sobre o problema e os requisitos. Consequentemente, quanto melhor o entendimento compartilhado, mais leve pode ser o processo de ER [GIFr2015].

Complexidade e criticidade. O grau de detalhamento no qual os requisitos precisam ser especificados depende fortemente da complexidade e criticidade do sistema a ser desenvolvido. Quando um sistema é complexo e/ou crítico em relação à segurança ou proteção, o processo de ER escolhido deve acomodar uma especificação detalhada dos requisitos críticos, incluindo modelos formais ou semiformais e forte

Complexidade e criticidade

validação - por exemplo, verificando modelos que expressam comportamento prescrito ou construindo protótipos.

Restrições. Obviamente, todos os fatores influenciadores restringem o espaço de possíveis configurações de um processo de ER. Quando falamos de restrições, referimo-nos às restrições impostas explicitamente, por exemplo, pelo cliente ou por um regulador.

Restrições

Tais restrições podem implicar na criação obrigatória de certos produtos de trabalho e seguir um processo obrigatório para produzir esses produtos de trabalho. Clientes ou reguladores também podem exigir um processo de RE que esteja em conformidade com um determinado padrão.

Tempo e orçamento disponível. Se os cronogramas e orçamentos forem apertados, o tempo e o orçamento disponíveis para RE precisam ser usados com sabedoria, o que normalmente implica na escolha de um processo leve de RE. A escolha de um processo de RE iterativo ajuda a priorizar os requisitos e implementar os mais importantes dentro do orçamento e cronograma fornecidos.

Tempo e orçamento disponível

Volatilidade dos requisitos. Se for provável que muitos requisitos mudem, é aconselhável escolher um processo de RE iterativo e amigável para mudanças.

Volatilidade dos requisitos

Experiência dos Engenheiros de Requisitos. O processo de ER escolhido deve corresponder às competências e experiência dos Engenheiros de Requisitos envolvidos. Caso contrário, tempo e orçamento adicionais devem ser alocados para treinar e treinar o processo escolhido.

Experiência de Engenheiros de Requisitos

É melhor escolher um processo bastante simples que os Engenheiros de Requisitos possam lidar adequadamente do que um sofisticado e complicado que os sobrecarregue.

5.2 Facetas do Processo de Engenharia de Requisitos

Definir o processo de RE do zero para cada empreendimento de RE é um desperdício de esforço. Sempre que os fatores influenciadores o permitam, o processo deve ser configurado a partir de elementos pré-existentes. Para fornecer orientação sobre como configurar um processo de RE adequado, descrevemos três facetas com duas instâncias cada, juntamente com os critérios de seleção a serem considerados para cada instância [Glin2019]. Posteriormente, na Seção 5.3, usaremos essas facetas para configurar os processos de ER. A Figura 5.1 mostra uma visão geral das facetas e instâncias.

Visão geral das facetas do processo

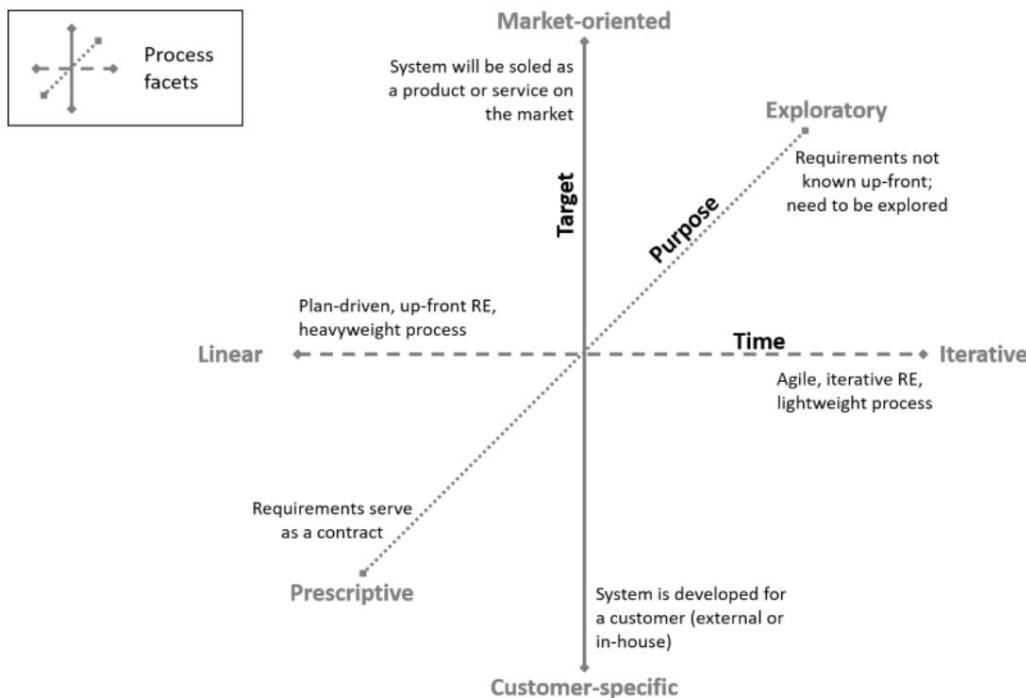


Figura 5.1 Facetas do processo de ER

As facetas podem ser consideradas como abrangendo um espaço tridimensional de opções de configuração do processo. Cada instância de faceta vem com critérios para selecioná-la.

A aplicabilidade desses critérios decorre da análise dos fatores de influência discutidos na Seção 5.1 acima. Observe que nem todos os critérios precisam ser atendidos para escolher uma instância de uma faceta.

5.2.1 Faceta do Tempo: Linear versus Iterativo

A faceta do tempo lida com a organização das atividades de ER em uma escala de tempo. Distinguimos entre processos lineares e iterativos.

Em um *processo de ER linear*, os requisitos são especificados antecipadamente em uma única fase do processo. A ideia é produzir uma especificação de requisitos abrangente que requer pouca ou nenhuma adaptação ou poucas mudanças durante o projeto e implementação do sistema. A criação antecipada de uma especificação de requisitos abrangente exige um processo abrangente. Assim, na maioria dos casos, os processos RE lineares são processos pesados.

Linear

Critérios para escolher um processo de RE linear:

- O processo de desenvolvimento do sistema é orientado a planos e principalmente linear.
- As partes interessadas estão disponíveis, conhecem seus requisitos e podem especificá-los antecipadamente.
- Uma especificação de requisitos abrangente é necessária como base contratual para a terceirização ou licitação do projeto e implementação do sistema.
- As autoridades reguladoras exigem uma especificação de requisitos abrangente e formalmente liberada em um estágio inicial do desenvolvimento.

Em um *processo de RE iterativo*, os requisitos são especificados de forma incremental, começando com objetivos gerais e alguns requisitos iniciais e, em seguida, adicionando ou modificando requisitos em cada iteração. A ideia é entrelaçar a especificação de requisitos com o projeto e implementação do sistema.

Iterativo

Devido aos ciclos de feedback curtos e à capacidade de acomodar mudanças ou coisas esquecidas em iterações posteriores, os processos de RE iterativos podem ser processos leves.

Critérios para escolher um processo de RE iterativo:

- O processo de desenvolvimento do sistema é iterativo e ágil.
- Muitos requisitos não são conhecidos de antemão, mas irão surgir e evoluir durante o desenvolvimento do sistema.
- As partes interessadas estão disponíveis de forma que curtos ciclos de feedback possam ser estabelecidos como um meio de mitigar o risco de desenvolver o sistema errado.
- A duração do desenvolvimento permite mais do que apenas uma ou duas iterações.
- A capacidade de alterar os requisitos facilmente é importante.

5.2.2 Faceta do Propósito: Prescritiva versus Explorativa

A faceta de propósito lida com o propósito e o papel dos requisitos no desenvolvimento de um sistema. Distinguimos entre processos de ER prescritivos e exploratórios.

Em um processo de *ER prescritivo*, a especificação de requisitos constitui um contrato: todos os requisitos são obrigatórios e devem ser implementados. A ideia é criar uma especificação de requisitos que possa ser implementada com pouca ou nenhuma interação adicional entre as partes interessadas e os desenvolvedores.

Prescritivo

Critérios para escolher um processo de RE prescritivo:

- O cliente exige um contrato fixo para o desenvolvimento do sistema, muitas vezes com funcionalidade, escopo, preço e prazo fixos.
- Funcionalidade e escopo têm precedência sobre custo e prazos.
- O desenvolvimento do sistema especificado pode ser licitado ou terceirizado.

Em um processo de *ER exploratório*, apenas os objetivos são conhecidos a priori, enquanto os requisitos concretos devem ser eliciados. A ideia é que os requisitos frequentemente não são conhecidos a priori, mas precisam ser explorados.

Explorador

Critérios para escolher um processo de RE exploratório:

- Inicialmente, as partes interessadas têm apenas uma vaga ideia sobre seus requisitos.
- As partes interessadas estão fortemente envolvidas e fornecem feedback contínuo.
- Prazos e custos têm precedência sobre funcionalidade e escopo.
- O cliente está satisfeito com um contrato de estrutura sobre metas, recursos e o preço a ser pago por um determinado período de tempo ou número de iterações.
- Não está claro a priori quais requisitos devem realmente ser implementados e em que ordem eles serão implementados.

5.2.3 Faceta Alvo: Específico ao Cliente versus Orientado para o Mercado

A faceta alvo considera o tipo de desenvolvimento: que tipo de desenvolvimento visamos com o processo de ER? Em um nível elementar, distinguimos entre processos de RE específicos do cliente e processos de RE orientados para o mercado.

Específico do cliente

Em um processo de *RE específico do cliente*, o sistema é encomendado por um cliente e desenvolvido por um fornecedor para esse cliente. Observe que o fornecedor e o cliente podem fazer parte da mesma organização. A ideia é que o processo de RE reflete a relação cliente-fornecedor.

Critérios para escolher um processo de RE específico do cliente:

- O sistema será usado principalmente pela organização que encomendou o sistema e paga pelo seu desenvolvimento.
- As partes interessadas importantes estão principalmente associadas à organização do cliente.
- Pessoas individuais podem ser identificadas para as funções das partes interessadas.
- O cliente deseja uma especificação de requisitos que possa servir como contrato.

Em um processo de *ER orientado para o mercado*, o sistema é desenvolvido como um produto ou serviço para um mercado, visando segmentos específicos de usuários. A ideia é que a organização que desenvolve o sistema também conduza o processo de ER.

Orientado para o mercado

Critérios para escolher um processo de RE orientado para o mercado:

- A organização desenvolvedora ou um de seus clientes pretende vender o sistema como um produto ou serviço em algum segmento de mercado.
- Os usuários em potencial não são identificáveis individualmente.
- Os Engenheiros de Requisitos devem projetar os requisitos para que correspondam às necessidades previstas dos usuários-alvo.
- Proprietários de produtos, pessoal de marketing, designers digitais e arquitetos de sistemas são os principais interessados.

5.2.4 Dicas e Advertências

É importante notar que os critérios dados acima são heurísticos. Eles não devem ser considerados como um conjunto de regras fixas que sempre se aplicam. Por exemplo, terceirizar o desenvolvimento do sistema é feito preferencialmente com um processo de ER prescritivo ao invés de exploratório. Isso ocorre porque o contrato entre o cliente e o fornecedor é normalmente baseado em uma especificação de requisitos abrangente.

Heurística, sem regras

No entanto, também é possível negociar um contrato de terceirização com base em um processo de RE exploratório.

Pode haver pré-requisitos para a escolha de certas instâncias de facetas do processo ou a escolha pode acarretar consequências que devem ser consideradas. aqui estão alguns exemplos:

Pré-requisitos e consequências

- Os processos de ER lineares funcionam apenas se um processo sofisticado para alterar os requisitos estiver em vigor.
- Processos lineares de ER implicam em longos ciclos de feedback: pode levar meses ou até anos desde a escrita de um requisito até que seus efeitos sejam observados no sistema implementado.
Para mitigar o risco de desenvolver o sistema errado, os requisitos devem ser validados intensivamente ao usar um processo de ER linear.
- Em um processo orientado para o mercado, o feedback dos usuários em potencial é o único meio de validar se o produto realmente atenderá às necessidades do segmento de usuários-alvo.
- Em um ambiente ágil, um processo de RE iterativo e exploratório se encaixa melhor.
As iterações têm uma duração fixa (normalmente de 2 a 6 semanas). O proprietário do produto desempenha um papel fundamental no processo de RE, coordenando as partes interessadas, organizando os produtos de trabalho de RE e comunicando os requisitos para o desenvolvimento equipe.

As três facetas mencionadas acima não são totalmente independentes: a escolha feita por uma faceta pode influenciar o que pode ou deve ser escolhido em outras. aqui estão alguns exemplos:

Influência mútua

- Linear e prescritivo são freqüentemente escolhidos juntos, o que significa que quando os Engenheiros de Requisitos decidem por um processo de RE linear, eles normalmente decidem por um processo que é tanto linear quanto prescritivo.
- Os processos exploratórios de ER são tipicamente também processos iterativos (e vice-versa).
- Um processo de ER orientado para o mercado não combina bem com um processo linear e prescritivo.

5.2.5 Considerações Adicionais

O grau em que um processo de ER deve ser estabelecido e seguido, bem como o volume de produtos de trabalho de requisitos a serem produzidos nesse processo, depende do grau de entendimento compartilhado e também da criticidade do sistema.

Compreensão compartilhada e criticidade

Quanto melhor o entendimento compartilhado e menor a criticidade, mais simples e leve pode ser o processo de ER.

Tempo e orçamento

Quando há pouco tempo e orçamento disponível para ER, os recursos disponíveis devem ser usados com cuidado. Escolher um processo iterativo e exploratório ajuda. Além disso, o processo deve se concentrar em identificar e lidar com os requisitos que são críticos para o sucesso do sistema.

Por fim, o processo de ER deve se adequar à experiência dos Engenheiros de Requisitos. Quanto menores forem suas habilidades e experiência, mais simples deve ser o processo de ER - não faz sentido definir um processo sofisticado quando as pessoas envolvidas não podem executar esse processo adequadamente.

Experiência de Engenheiros de Requisitos

5.3 Configurando um Processo de Engenharia de Requisitos

Em um contexto de desenvolvimento de sistema concreto, os Engenheiros de Requisitos ou a(s) pessoa(s) responsável(is) pelo RE devem escolher o processo de RE a ser aplicado. Recomendamos analisar os fatores de influência (consulte a Seção 5.1) primeiro e, em seguida, selecionar uma combinação adequada das facetas do processo descritas na Seção 5.2.

Configurações do processo

5.3.1 Combinações Típicas de Facetas

Três combinações de facetas (ou variantes delas) ocorrem frequentemente na prática [Glin2019]. A seguir, descrevemos brevemente cada um deles e os caracterizamos em termos de seu principal caso de aplicação, produtos de trabalho típicos e fluxo de informações típico. Além disso, fornecemos um exemplo. A Figura 5.2 mostra as três configurações típicas de processo no espaço das três facetas.

Processo de RE participativo: iterativo e exploratório e específico do cliente

Um processo de ER participativo é normalmente escolhido em configurações ágeis quando há um cliente que solicita um sistema e uma equipe de desenvolvimento que o projeta e implementa. O foco é explorar os requisitos em uma série de iterações em estreita colaboração entre as partes interessadas do lado do cliente, os Engenheiros de Requisitos e a equipe de desenvolvimento.

Participativo | Principalmente ágil | Para um cliente

Caso de aplicação principal: Fornecedor e cliente colaboram estreitamente; as partes interessadas estão fortemente envolvidas nos processos de ER e de desenvolvimento.

Produtos de trabalho típicos: Product backlog com histórias de usuários e/ou descrições de tarefas, visão, protótipos

Fluxo de informação típico: Intereração contínua entre partes interessadas, proprietários de produtos, engenheiros de requisitos e desenvolvedores

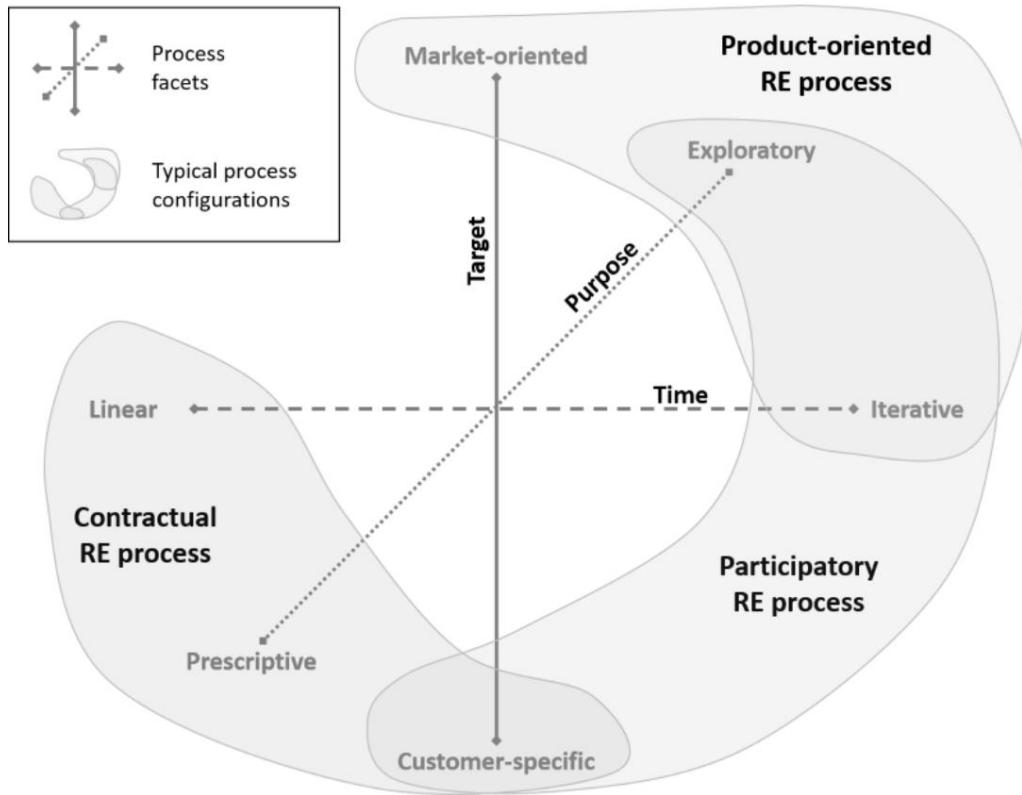


Figura 5.2 Três configurações típicas de processo de ER e sua relação com as três facetas

Exemplo: Em uma seguradora, a unidade de negócios que vende seguros corporativos para pequenas e médias empresas tem uma ideia sobre um novo produto para proteger os clientes contra danos causados por um ataque de hackers. Eles contratam a unidade corporativa de TI da empresa para formar uma equipe de desenvolvimento com a tarefa de projetar e desenvolver um novo aplicativo que possa lidar com o novo produto de seguro dentro do sistema de suporte de vendas de seguros existente. Além disso, o sistema de gerenciamento de contrato de seguro existente precisa ser adaptado de acordo. Além de alguns requisitos iniciais, a unidade de negócios contratante não tem uma ideia clara de como o novo produto deve ser e como deve ser suportado pelos sistemas corporativos de TI. A TI corporativa adotou o desenvolvimento ágil para todos os seus projetos há alguns anos.

Exemplo: Apoiar um novo produto de seguro

Nesta situação, um processo de ER participativo é apropriado. Ele se encaixa no processo ágil geral que a TI corporativa empregará para desenvolver o novo sistema e adaptar os existentes. As partes interessadas da unidade de negócios e os engenheiros de requisitos da TI corporativa podem obter em conjunto os requisitos para o novo produto de seguro. Como o processo é iterativo, a equipe de desenvolvimento pode desenvolver um protótipo de produto mínimo comercializável (MMP) que ajuda a gestão da unidade de negócios a decidir se inclui ou não o produto previsto em seu portfólio ou descarta a ideia.

Existe uma clara relação cliente-fornecedor entre a unidade de negócios e a TI corporativa, portanto, um processo de RE orientado para o cliente se encaixa.

Processo de RE Contratual: Tipicamente Linear e Prescritivo e Específico do Cliente

Um processo de RE contratual é normalmente escolhido quando o desenvolvimento de um sistema é licitado e terceirizado para um provedor com um contrato baseado em uma especificação de requisitos abrangente. Também é um processo adequado para RE em grandes projetos de desenvolvimento de sistemas que aplicam um processo de desenvolvimento em estilo cascata.

*Contratual | Antecipado
RE | Cachoeira |
Para um cliente*

Caso de aplicação principal:	A especificação de requisitos constitui a base contratual para o desenvolvimento de um sistema por pessoas não envolvidas na especificação e com pouca interação das partes interessadas após a fase de requisitos.
Produtos de trabalho típicos:	Especificação clássica de requisitos do sistema, consistindo em requisitos textuais e modelos
Fluxo de informação típico:	Principalmente das partes interessadas para os Requisitos engenheiros

Exemplo: Um fabricante de automóveis está desenvolvendo uma nova plataforma de automóveis, da qual será derivada uma família de modelos de automóveis. Uma importante decisão de design para a nova plataforma é livrar-se das dezenas de unidades de controle eletrônico (ECUs) atualmente usadas nos carros e substituí-las por um único computador de controle que executa uma pilha de aplicativos de controle e assistência à direção. O objetivo é economizar custos de hardware, livrar-se de interações indesejadas entre ECUs e reduzir o tempo e o esforço para realizar atualizações do software. Os engenheiros responsáveis pelos sistemas eletrônicos da nova plataforma redigiram uma especificação de requisitos do cliente. A empresa contratou um grande fabricante de sistemas de controle automotivo para criar uma especificação de requisitos de sistema para o novo sistema de controle centralizado de carros. Posteriormente, o fabricante do automóvel licitará o projeto e a implementação do sistema com base nessa especificação. O fabricante exigirá que a implementação seja realizada em várias iterações para facilitar os testes e a integração do sistema com a nova plataforma do carro.

Exemplo: Especificação de um novo sistema de controle na indústria automotiva

Nesta situação, um processo de RE contratual é apropriado. O processo geral é linear: o sistema será projetado e implementado somente após a conclusão da especificação dos requisitos. O fato de a implementação ser iterativa não impacta o processo de ER. Dependendo da qualidade da especificação de requisitos do cliente existente e da disponibilidade das partes interessadas no fabricante do carro, um processo de RE linear ou iterativo deve ser escolhido.

Obviamente, é necessário um processo de RE orientado para o cliente. A existência de uma especificação de requisitos do cliente e o fato de que a especificação de requisitos do sistema será usada para licitar o projeto e a implementação da chamada do sistema para um processo de RE prescritivo.

Processo de RE orientado para o produto: iterativo e exploratório e orientado para o mercado

Um processo de RE orientado para o produto é normalmente escolhido quando uma organização está desenvolvendo um sistema como um produto ou serviço para o mercado. Na maioria dos casos, um processo de RE orientado para o produto vem junto com um processo ágil de desenvolvimento do produto. O proprietário do produto e os designers digitais desempenham papéis importantes nesse processo: eles influenciam e moldam fortemente o produto.

Orientado para o produto | Principalmente ágil | para o mercado

Caso de aplicação principal:	Uma organização especifica e desenvolve software para vendê-lo ou distribuí-lo como um produto ou serviço
Produtos de trabalho típicos:	Backlog do produto com histórias de usuários e/ou descrições de tarefas, visão, protótipos, feedback do usuário
Fluxo de informação típico:	Interação entre o proprietário do produto, marketing, Engenheiros de requisitos, designers digitais e desenvolvedores, além de feedback de clientes/usuários

Exemplo: renovação total de um aplicativo móvel de notícias

Exemplo: uma empresa de mídia encarrega sua TI interna de renovar totalmente o aplicativo móvel de notícias que a empresa vende para assinantes (com alguns conteúdos sendo de acesso gratuito). A partir do feedback do usuário, a empresa mantém um longo registro de críticas de clientes e sugestões de melhoria.

Em particular, muitos usuários criticam o aplicativo existente por não ser responsivo o suficiente, por ter um suporte ruim para relatar problemas e sugestões e por não oferecer zoom com dois dedos em texto ou imagens. O departamento de marketing da empresa também percebe que o layout do aplicativo está desatualizado. Eles preveem que, com um novo layout, mais assinantes podem ser conquistados. O CEO da empresa decidiu que o departamento de TI deve colaborar com uma agência de design externa para a aparência visual do aplicativo. A administração da empresa deseja uma versão mínima do produto como prova de conceito e, em seguida, novas versões intermediárias a cada três semanas que possam ser revisadas pelo departamento de marketing e pela diretoria da empresa.

Nesta situação, um processo de RE orientado para o produto se encaixa melhor. Embora exista uma relação cliente-fornecedor entre a gestão da empresa e o seu departamento informático interno, o foco está claramente na criação de um produto renovado no segmento das aplicações móveis de notícias. O processo de ER precisa ser exploratório, pois os requisitos além das informações no log existente de feedback do usuário não são claros. O processo geral de desenvolvimento deve ser iterativo de acordo com a decisão da administração da empresa. Como os requisitos precisam ser explorados, um processo de RE iterativo é o mais adequado aqui.

5.3.2 Outros Processos de RE

As três combinações descritas acima abrangem muitas das situações que ocorrem na prática. No entanto, pode haver situações em que nenhuma das configurações de processo mencionadas anteriormente se encaixe. Por exemplo, restrições regulatórias podem impor o uso de um processo que esteja em conformidade com um determinado padrão, como ISO/IEC/IEEE 29148 [ISO29148].

Casos especiais precisam de processos especiais de RE

Nesse caso, o processo de ER deve ser criado por especialistas do processo a partir do zero ou uma das configurações mencionadas acima deve ser adaptada para que seja adaptada à situação em questão.

5.3.3 Como Configurar Processos de RE

Recomendamos um procedimento de cinco etapas para configurar um processo de RE.

Um procedimento de configuração de cinco etapas

1. *Analise os fatores influenciadores.* Analise sua situação com relação à lista de fatores de influência da Seção 5.1.
2. *Avalie os critérios de faceta.* Com base na análise da etapa 1, percorra a lista de critérios de seleção de faceta fornecida na Seção 5.2. Você pode atribuir a cada critério um valor em uma escala de cinco pontos (—, -, 0, +, ++).
3. *Configurar.* Se a análise de critérios produzir um resultado claro com relação às três configurações típicas mencionadas acima, escolha essa configuração. Caso contrário, escolha uma adaptação de processo diferente, guiada pelo objetivo geral de mitigar o risco de desenvolver o sistema errado. Por exemplo, imagine uma situação em que o cliente exige que uma especificação de requisitos do sistema seja criada antecipadamente, o que exige um processo de ER linear e prescritivo. No entanto, em suas primeiras reuniões com o cliente, você notou que, para um subsistema importante, o cliente não tem uma ideia clara do que construir, o que exige um processo de RE exploratório. Uma solução potencial poderia ser escolher um processo de RE contratual como a estrutura geral do processo de RE, mas criar um

subprojeto que elicia passo a passo os requisitos para aquele subsistema importante, criando protótipos em duas ou três iterações (guiado por um subprocesso de ER participativo) e, em seguida, alimentando os resultados na especificação de requisitos do sistema.

4. Determine os produtos de trabalho. Com base em sua análise e configuração do processo, defina os principais produtos de trabalho de RE que serão produzidos. Certifique-se de que os produtos de trabalho de RE estejam alinhados com os produtos de trabalho do processo de desenvolvimento geral.

5. Selecione as práticas apropriadas. Para as tarefas a serem executadas - por exemplo, levantamento de requisitos - selecione as práticas que melhor se encaixam na situação em questão. Muitas dessas práticas, incluindo dicas sobre onde e quando aplicá-las, são apresentadas nos Capítulos 2, 4 e 6 deste manual.

Não existe um processo de RE comprovado e único para todos. Com base em uma análise dos fatores de influência, um processo de RE específico precisa ser adaptado para cada empreendimento de RE. Uma forma simples de adaptação é configurar um processo de ER a partir de um conjunto de facetas de processo.

5.4 Leitura Adicional

Armour [Armo2004] e Reinertsen [Rein1997], [Rein2009] fornecem reflexões gerais sobre processos e fluxos de informações em processos.

Embora o livro-texto de Robertson e Robertson [RoRo2012] seja intitulado “Dominando o Processo de Requisitos”, este é um livro-texto geral sobre todos os aspectos de ER.

Wiegers e Beatty [WiBe2013] fornecem um capítulo sobre como melhorar os processos de ER.

O livro de Sommerville e Sawyer [SoSa1998] contém uma coleção de boas práticas a serem usadas na estrutura dos processos de ER.

6 Práticas de Gestão de Requisitos

Os requisitos não são esculpidos em pedra, eternamente presentes do passado para o futuro; eles estão vivos! Eles nascem através da elicitação, crescem através da documentação e são moldados através da validação. Quando adultos, eles vão trabalhar por meio da implementação e depois de uma vida longa e próspera em operação, se aposentam no esquecimento.

Ao longo de seu ciclo de vida, seus pais, os Engenheiros de Requisitos, cuidam deles. Cuidamos deles na infância, os ensinamos na juventude, os acompanhamos em seus relacionamentos e os ajudamos a encontrar um bom emprego em um sistema saudável. Isso é o que chamamos de gerenciamento de requisitos.

Claro, existem definições melhores e mais formais de gerenciamento de requisitos.

O padrão ISO/IEC/IEEE 29148:2018 [ISO29148] define o gerenciamento de requisitos como "*atividades que identificam, documentam, mantêm, comunicam, rastreiam e rastreiam requisitos ao longo do ciclo de vida de um sistema, produto ou serviço*". No glossário CPRE [Glin2020], o gerenciamento de requisitos é definido como "*O processo de gerenciamento de requisitos existentes e produtos de trabalho relacionados a requisitos, incluindo o armazenamento, alteração e rastreamento de requisitos*". O glossário CPRE também nos diz que *o gerenciamento de requisitos* é parte integrante da Engenharia de Requisitos: "*A abordagem sistemática e disciplinada para a especificação e gerenciamento de requisitos com o objetivo de ...*".

O gerenciamento de requisitos pode ocorrer em diferentes níveis:

- Os requisitos individuais
- Os produtos de trabalho que contêm esses requisitos
- O sistema relacionado aos produtos de trabalho e os requisitos neles contidos

O gerenciamento de requisitos ocorre em diferentes níveis

Na prática, o gerenciamento de requisitos é realizado principalmente no nível do produto de trabalho. Normalmente, um produto de trabalho contém vários requisitos individuais (por exemplo, uma descrição de interface externa), enquanto outros produtos de trabalho contêm apenas um único requisito (por exemplo, uma única história de usuário em um projeto ágil) ou representam todo o conjunto de requisitos para um sistema (por exemplo, software especificação de requisitos). Esteja ciente de que todos os produtos de trabalho de todos os três níveis devem ser gerenciados e certifique-se de conhecer as relações entre eles.

O texto acima descreve o *que* é gerenciamento de requisitos. O restante deste capítulo é dedicado ao *como*: todos os tipos de práticas aplicáveis para fazer o gerenciamento de requisitos funcionar.

o que versus como

Antes de mergulharmos nos detalhes do gerenciamento de requisitos, vamos considerar alguns princípios básicos para fazê-lo funcionar. Se você deseja gerenciar algo, deve ser capaz de reconhecê-lo, armazená-lo e encontrá-lo novamente.

Portanto, identificação exclusiva, um grau apropriado de padronização, prevenção de redundância, um repositório central e acesso gerenciado são essenciais.

Na Seção 6.1, examinamos brevemente as situações que influenciam o valor, a importância e o esforço envolvidos no gerenciamento de requisitos.

A Seção 6.2 segue os requisitos em seu ciclo de vida como parte dos produtos de trabalho que os Engenheiros de Requisitos e outras equipes de TI produzem e usam durante o desenvolvimento, implementação e operação de um sistema de TI.

Durante o ciclo de vida de um requisito, várias versões de produtos de trabalho (e os requisitos que eles contêm) são criadas, começando com um rascunho 0.1 inicial que, após uma série de alterações maiores e menores, evolui para, digamos, uma versão final 3.2. O controle de versão é discutido na Seção 6.3.

Ao desenvolver e usar sistemas de TI, é impraticável lidar com todos os requisitos individualmente. Portanto, conjuntos coerentes de requisitos são reconhecidos como configurações e linhas de base, conforme explicado na Seção 6.4.

Para lidar com produtos de trabalho e requisitos de forma eficiente, devemos ser capazes de identificá-los e coletar dados sobre eles. Esse é o tópico da Seção 6.5.

A Seção 6.6 analisa a rastreabilidade dos requisitos. A rastreabilidade é uma característica de qualidade especialmente importante dos requisitos, como você já deve ter entendido ao ler as definições de gerenciamento de requisitos acima. Sem rastreabilidade, é impossível vincular o comportamento real de um sistema às demandas originais das partes interessadas.

A Seção 6.7 trata das mudanças nos requisitos que ocorrem durante sua vida útil. Nas primeiras fases de sua existência, as mudanças podem ser frequentes, mas após a validação, os requisitos devem ser estáveis. No entanto, as mudanças ainda ocorrerão. Para aplicá-los de maneira ordenada, um processo definido para lidar com a mudança deve estar em vigor.

Por natureza, os requisitos diferem em importância e valor. Normalmente, os recursos para elaborá-los são limitados, portanto nem todos os requisitos chegarão à implementação.

Isso significa que as partes interessadas terão que decidir quando um determinado requisito será implementado ou mesmo se será implementado ou não. A priorização, descrita na Seção 6.8, pode sustentar essa decisão.

6.1 O que é Gerenciamento de Requisitos?

Na introdução, já vimos que gerenciamento de requisitos significa o gerenciamento de requisitos existentes e produtos de trabalho relacionados a requisitos, incluindo armazenamento, alteração e rastreamento dos requisitos. Mas por que gerenciá-los?

Gerenciamos os requisitos porque são coisas vivas; eles são criados, usados, atualizados e excluídos novamente durante seu desenvolvimento e operação. E durante todo esse ciclo de vida, devemos garantir que todas as partes envolvidas tenham acesso às versões corretas de todos os requisitos relevantes para elas. Se não gerenciarmos os requisitos adequadamente, corremos o risco de que algumas partes possam ignorar requisitos, aderir a requisitos desatualizados, trabalhar com versões erradas, negligenciar relacionamentos e assim por diante. Isso pode prejudicar seriamente a eficiência e eficácia do desenvolvimento e uso do sistema. Em outras palavras: o valor do gerenciamento adequado de requisitos reside na melhoria da eficiência e eficácia de um sistema.

Isso significa que o valor do gerenciamento de requisitos não pode ser separado do valor do sistema em questão e seu contexto. Na prática, podemos ver grandes diferenças na importância e no nível do gerenciamento de requisitos e no esforço envolvido [Rupp2014], desde uma tarefa subsidiária informal de um Engenheiro de Requisitos com uma planilha, até uma função em tempo integral de um gerente de requisitos dedicado com um banco de dados de requisitos suportado por ferramentas.

Valor da gestão de requisitos

É necessário um gerenciamento de requisitos mais completo com um número maior de requisitos, partes interessadas e desenvolvedores, com uma expectativa de vida mais longa, mais mudanças ou demandas de qualidade mais altas no sistema e com um processo de desenvolvimento mais complexo, padrões, normas e regulamentos mais rígidos , incluindo a necessidade de uma trilha de auditoria detalhada.

Frequentemente, vemos que o gerenciamento de requisitos é um pouco negligenciado no início de um projeto, quando uma pequena equipe está trabalhando em um conjunto óbvio de requisitos de alto nível. Posteriormente, a complexidade aumenta e a equipe perde a visão geral, resultando em problemas de qualidade e redução da eficiência. Então, muito esforço deve ser gasto para alcançar o nível de controle necessário. É mais eficiente investir algum esforço desde o início de um projeto para configurar os recursos e processos de gerenciamento de requisitos com as demandas esperadas no final.

6.2 Gestão do Ciclo de Vida

Conforme declarado na introdução, requisitos e produtos de trabalho que contêm requisitos têm uma vida útil. Nós os vemos sendo criados, elaborados, validados, consolidados, implementados, usados, alterados, mantidos, retrabalhados, refatorados, retirados, arquivados e/ou excluídos. Isso é o que queremos dizer com seu ciclo de vida: durante sua vida, um requisito pode estar em um número limitado de estados e pode mostrar um número limitado de transições de estado com base em eventos explícitos no contexto. A Figura 6.1 mostra um *diagrama de estado* simplificado como um modelo para o ciclo de vida de um único requisito (somente visão geral, as transições de estado não são mostradas; por exemplo, a transição do estado composto *Em desenvolvimento* para *Em produção* pode ser acionada por uma decisão de ativação do proprietário do produto).

Gerenciamento do ciclo de vida de requisitos e produtos de trabalho

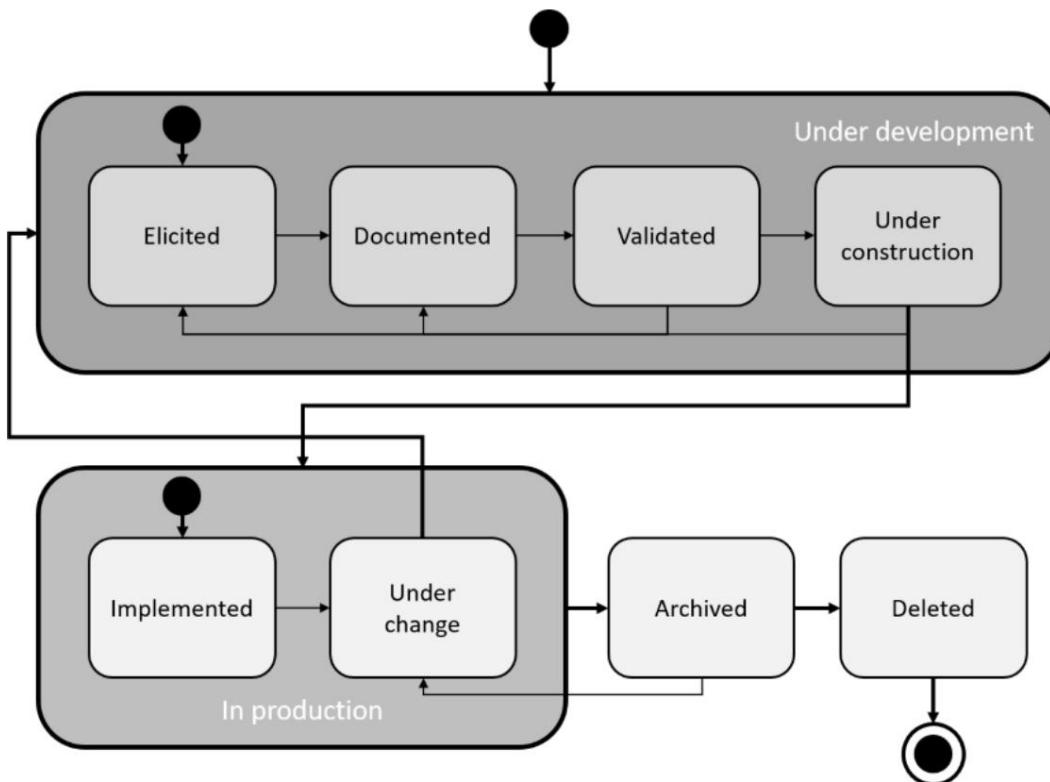


Figura 6.1 Gráfico de estados simplificado de um ciclo de vida de requisitos

Um fator complicador é que os produtos de trabalho e os requisitos individuais têm seus próprios ciclos de vida diferentes que se sobrepõem apenas parcialmente. Como exemplo, pense em um *estudo de definição* de produto de trabalho no estado *em mudança*; isso não significa necessariamente que todos os requisitos contidos no produto de trabalho devam ser alterados. E para o mesmo *estudo de definição*, o estado *implementado* não faz sentido; apenas alguns requisitos nele serão implementados - ou melhor: determinado código, baseado nesses requisitos.

Outro fator complicador pode ser que, na prática, a visão do ciclo de vida dos requisitos é diferente para diferentes funções. Para você como um Engenheiro de Requisitos, para rastrear seu trabalho, você está interessado em diferentes estados para o gerente de projeto, e outros estados novamente em comparação com o gerente de produto ou um gerente de mudanças: no diagrama acima, seu interesse pode terminar em validado, enquanto para o gerente de projeto, ele só começa no documentado.

Os Engenheiros de Requisitos gerenciam ativamente o ciclo de vida de seus produtos de trabalho. A gestão do ciclo de vida implica:

- Definir modelos de ciclo de vida para seus produtos de trabalho e os requisitos contidos neles com
- Os estados que um produto de trabalho ou requisito pode levar
- As transições permitidas entre esses estados
- Os eventos que desencadeiam a transição de um estado para outro
- Garantir que ocorram apenas transições explicitamente permitidas
- Registrar os estados reais que os produtos de trabalho e requisitos levam
- Registrando as transições reais que ocorrem
- Relatórios sobre esses estados e transições

O gerenciamento do ciclo de vida precisa ser gerenciado ativamente

Em palavras simples: certifique-se de saber o estado em que seus requisitos estavam, estão e ficarão, como eles podem mudar e por que tudo isso acontece.

Por exemplo, como Engenheiro de Requisitos, você pode ser solicitado a relatar quem aprovou qual versão de um requisito a ser lançado como entrada para a fase de codificação. Acompanhar os estados dos requisitos em seu ciclo de vida também pode ser útil para criar painéis e relatar o andamento de um projeto. Pode ser uma boa maneira de organizar o trabalho e identificar em quais requisitos trabalhar primeiro.

O estado de um produto de trabalho sob o gerenciamento do ciclo de vida geralmente é registrado em um atributo (consulte a Seção 6.5). Também pode ser útil documentar a data inicial e final desse estado em atributos. Em projetos ágeis, o estado de um produto de trabalho (item) pode ser derivado de sua posição no backlog do produto, backlog de tarefas e/ou no quadro de tarefas. Além disso, atender aos critérios da definição de pronto e da definição de concluído pode fornecer informações relevantes, pois atender a esses critérios significa, na verdade, atingir um próximo estado.

A minúcia e o nível de detalhamento do gerenciamento do ciclo de vida devem ser adaptados às necessidades do cliente, do projeto e do sistema. Por exemplo, os estados em desenvolvimento, em produção e arquivados podem ser suficientes. Em projetos complexos ou críticos, você pode precisar de um modelo muito mais detalhado dos estados, procedimentos rígidos sobre transições de estado e uma trilha de auditoria que mostre o que aconteceu durante o projeto.

6.3 Controle de Versão

É comum que ambos, produtos de trabalho e requisitos individuais como parte de um produto de trabalho, sofram certas alterações durante seu ciclo de vida (consulte a Seção 6.7 para obter mais informações sobre como lidar com essas alterações). Após cada mudança, o produto de trabalho é diferente do que era antes: tornou-se uma nova versão.

Queremos controlar as versões desses produtos de trabalho por dois motivos:

- Às vezes, as mudanças dão errado. Depois de um tempo, defeitos são encontrados ou os benefícios pretendidos não são percebidos. Nesse caso, podemos implementar novas alterações em uma próxima versão, mas também podemos decidir voltar para uma versão anterior e continuar a partir daí. Ou talvez, pensando bem, preferimos a versão anterior, afinal.
- Queremos conhecer a história do produto do trabalho, entender sua evolução desde sua origem até sua situação atual. Isso pode nos ajudar quando tivermos que decidir sobre mudanças futuras ou apenas responder a perguntas sobre por que o produto de trabalho atual é o que é.

Razões para controle de versão de produtos de trabalho

O controle de versão requer três medidas a serem implementadas:

- Uma *identificação* de cada versão, para distinguir entre as diferentes versões de um produto de trabalho. Este é o número da versão, geralmente complementado com uma data de versão.
- Uma descrição clara de cada *alteração*. Você deve ser capaz de dizer - e entender - a diferença entre uma determinada versão e sua predecessora. Essa descrição de alteração deve estar claramente vinculada ao número da versão.
- Uma política rígida de *armazenamento* de versões, permitindo que você localize e recupere versões antigas. A menos que as limitações de armazenamento determinem o contrário, você deve preservar todas as versões anteriores de todos os seus produtos de trabalho; caso contrário, talvez não seja possível restaurar uma versão, se necessário. Por outro lado, o armazenamento ilimitado raramente será o caso, portanto, é aconselhável também ter uma política de arquivamento e limpeza de produtos de trabalho que não são mais usados.

Medidas para controle de versão

Normalmente, um produto de trabalho contém vários requisitos. Se um único requisito nesse produto de trabalho for alterado, tanto esse requisito quanto o produto de trabalho devem obter um novo número de versão, enquanto os requisitos inalterados nesse produto de trabalho mantêm seu número de versão antigo. Isso pode se tornar muito confuso em breve. Uma solução prática pode ser fazer a numeração de versão apenas no nível do produto de trabalho e deixar que todos os requisitos herdem o número da versão e o histórico de alterações do produto de trabalho.

Os números de versão são normalmente compostos de (pelo menos) duas partes:

Números de versão

- *Versão*. Em princípio, a versão começa em zero enquanto o produto de trabalho estiver em desenvolvimento. Quando for formalmente aprovado, liberado e/ou lançado, atribuímos a ele a versão *um*. Depois disso, a versão é aumentada apenas para atualizações importantes e substanciais.
- *Incremento*. Isso geralmente começa em *um* e é incrementado com cada alteração (externamente visível), no lado do conteúdo ou geralmente apenas textual ou editorial. Um sub-incremento adicional pode ser usado apenas para correção de erros de digitação. O incremento *nove às vezes* é usado para denotar uma versão final pouco antes da aprovação ou lançamento.

Um novo número de versão é atribuído a cada alteração formal.

Freqüentemente, uma mudança no estado do ciclo de vida de um produto de trabalho não é considerada uma razão para incrementar o número da versão, a menos que seja acompanhada por uma mudança no conteúdo ou no texto. Se, por exemplo, um requisito receber o estado validado e o número da versão 1.0 após a aprovação, não há necessidade de alterar esse número da versão se o estado mudar para em construção e posteriormente implementado. O estado pode finalmente terminar em arquivado, mas ainda manter o mesmo número de versão 1.0.

6.4 Configurações e linhas de base

Suponha que você preserve, conforme aconselhado acima, todas as versões de todos os requisitos que você desenvolve durante um projeto. Você terá então um banco de dados em constante expansão, repleto de requisitos e começará a perder a visão geral. Um dia, seu cliente chega à sua mesa e pergunta: "Implementamos seu sistema em todas as nossas agências. Agora parece haver um problema com os cálculos em nosso escritório de Barcelona. Você pode me dizer qual versão dos requisitos de cálculo eles usam lá?" Se você não puder responder a essa pergunta, desejará ter prestado mais atenção ao gerenciamento de configuração.

Então, o que é uma configuração? Você encontrará uma definição no glossário CPRE [Glin2020], mas em resumo, para um Engenheiro de Requisitos, uma configuração é um conjunto consistente de produtos de trabalho logicamente relacionados que contêm requisitos. Selecionamos esse conjunto com uma finalidade específica, geralmente para deixar claro quais requisitos são ou foram válidos em determinada situação.

Isso define as seguintes propriedades para uma configuração correta:

- *Logicamente conectado*. O conjunto de requisitos na configuração pertence em vista de um determinado objetivo.
- *Consistente*. O conjunto de requisitos não possui conflitos internos e pode ser integrado em um sistema.
- *Exclusivo*. Tanto a própria configuração quanto seus requisitos constituintes são identificados de forma clara e exclusiva.
- *Imutável*. A configuração é composta por requisitos selecionados, cada um com uma versão específica que nunca será alterada nesta configuração.
- *Base para reset*. A configuração permite fallback para uma configuração anterior se qualquer mudança indesejada parecer ter ocorrido.

[Configuração](#)

[Propriedades para configurações](#)

Uma configuração é documentada como um produto de trabalho, com uma identificação exclusiva, um estado e um número de versão e data, assim como qualquer outro produto de trabalho. No entanto, como uma configuração é por definição imutável, ela sempre terá apenas uma versão (por exemplo, 1.0).

Uma configuração sempre tem duas dimensões [CoWe1998]:

[Dimensões das configurações](#)

- A dimensão do produto .

Isso indica quais requisitos estão incluídos nessa configuração específica.

Às vezes, uma configuração conterá todos os requisitos disponíveis, mas geralmente é uma determinada seleção - por exemplo, todos os requisitos implementados na versão francesa de um sistema. O lançamento britânico do mesmo sistema pode ter uma configuração diferente.

► A dimensão da versão .

Em uma configuração específica, cada requisito selecionado está presente em exatamente uma, e apenas uma, versão. Pode ser a versão mais recente ou anterior, dependendo da finalidade da própria configuração. Assim que uma única versão diferente de um único requisito é selecionada, esta é uma nova configuração. Imagine um sistema para o qual será implementado um novo release com alguns requisitos em uma versão superior: esse novo release terá então uma configuração diferente.

A Figura 6.2 fornece outro exemplo de diferentes configurações que consistem em conjuntos específicos de versões de requisitos.

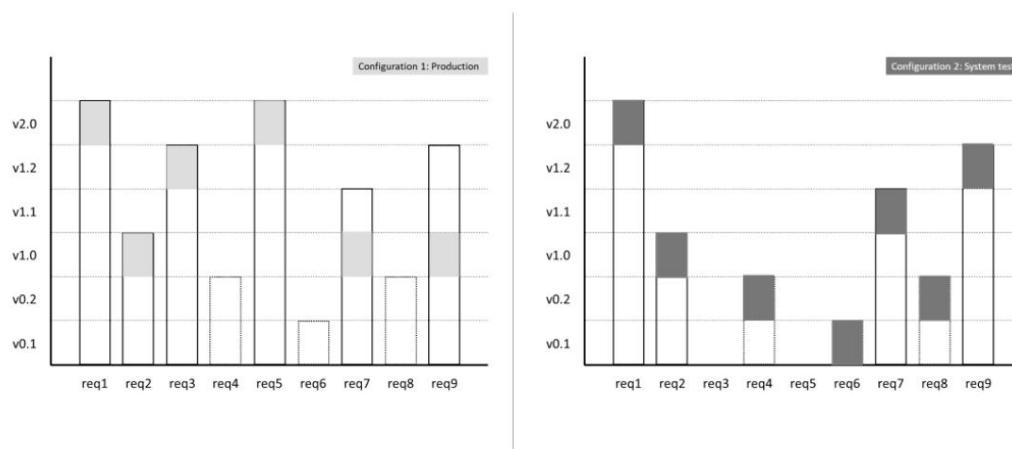


Figura 6.2 Exemplo de configurações

A figura acima mostra um exemplo de diferentes configurações de um determinado sistema. Ele mostra uma coleção de nove requisitos. Alguns deles ainda estão nos estágios iniciais de desenvolvimento – por exemplo, requisito 6 com a versão v0.1. Outros requisitos tiveram mais versões – por exemplo, o requisito 1, que está finalizado e já teve uma grande atualização, agora é a versão v2.0.

A imagem à esquerda mostra a configuração que está atualmente em produção. Ele consiste em R1 v2.0, R2 v1.0, R3 v1.2 (esse requisito teve duas pequenas atualizações após a implementação), R5 v2.0, R7 v1.0 e R9 v1.0. R4, R6 e R8, estando em desenvolvimento, não estão presentes nesta configuração, nem as novas versões de R7 e R9.

A figura da direita mostra a configuração que, ao mesmo tempo, está presente no ambiente de teste do sistema. Alguns requisitos (R1, R2) são os mesmos, alguns não estão mais presentes (R3, R5), os requisitos em desenvolvimento (R4, R6 e R8) estão incluídos aqui e dois requisitos (R7 e R9) estão presentes em um versão superior do que na configuração do ambiente de produção.

Em muitos projetos algumas configurações são tratadas de forma especial: essas configurações são chamadas de baselines. Uma *linha de base* é uma configuração estável, validada e controlada por alterações que marca um marco ou outro tipo de *ponto de repouso* no projeto. Um exemplo pode ser a configuração no final da fase de design, pouco antes de iniciar a fase de codificação, ou a configuração que é válida no go-live de um determinado lançamento.

Linha de base

O sprint backlog em um projeto ágil serve como linha de base no início da próxima iteração. As linhas de base são úteis para fins de planejamento, pois representam um ponto de partida estável para uma próxima fase. Eles geralmente são congelados e deixados de lado como uma âncora na vida agitada de um projeto. Se algo der muito errado no projeto, a equipe pode reverter para a situação da linha de base e reiniciar a partir daí.

Para o Engenheiro de Requisitos, é principalmente a configuração de produtos de trabalho contendo requisitos que é importante. Mas, na prática, a configuração dentro de um projeto tem um escopo muito mais amplo, contendo versões selecionadas dos produtos de trabalho de todos os membros da equipe, como requisitos, designs, código e casos de teste. Em projetos complexos, o gerenciamento de configuração pode ser um trabalho em tempo integral, executado com ferramentas dedicadas.

Não apenas requisitos

6.5 Atributos e Visualizações

Como Engenheiro de Requisitos, sua saída consiste em todos os tipos de produtos de trabalho que contêm requisitos. Esses requisitos terão que ser gerenciados, caso contrário, você e sua equipe perderão rapidamente a visão geral. Para gerenciar os requisitos, você precisa coletar e manter dados sobre eles – metadados, dados sobre dados. Os metadados tornam os produtos de trabalho tangíveis e gerenciáveis; por meio de metadados, você pode fornecer e obter informações sobre os requisitos e responder a perguntas relevantes durante e após o ciclo de vida do projeto ou produto. Pense em perguntas como “*Quais requisitos estão planejados para o próximo lançamento?*” ou “*Quanto esforço esse lançamento provavelmente exigirá?*” ou “*Quantos requisitos têm alta prioridade?*”

Ao considerar os requisitos como entidades sobre as quais a informação é requerida, as características desses requisitos são chamadas de *atributos*. Neste capítulo, já vimos alguns atributos comuns, como a identificação única, número da versão, estado, várias datas. Os atributos a serem definidos para os requisitos dependem das necessidades de informação dos stakeholders do projeto e do sistema.

Atributos

No início de um projeto, um *esquema de atributos* deve ser definido para permitir que o Engenheiro de Requisitos atenda a essas necessidades.

Um bom ponto de partida pode ser encontrado nas normas relevantes. O padrão ISO [ISO29148] menciona:

- ▶ *Identificação.* Cada requisito deve ter um identificador único e imutável, como um número, nome, mnemônico. Sem uma identificação adequada, o gerenciamento de requisitos é impossível.
- ▶ *Prioridade das partes interessadas.* A prioridade (acordada) do requisito do ponto de vista das partes interessadas. Consulte a Seção 6.8 para obter informações sobre como determinar essa prioridade.
- ▶ *Dependência.* Às vezes, há uma dependência entre os requisitos. Isso pode significar que um requisito de baixa prioridade deve ser implementado primeiro porque outro requisito de alta prioridade depende dele.

- ▶ **Risco.** Trata-se do potencial que a implementação do requisito levará a problemas, como danos, custos extras, atrasos, reclamações legais. Por natureza, trata-se de uma estimativa, baseada no consenso entre as partes interessadas.
- ▶ **Fonte.** Qual é a origem do requisito, de onde veio? Você pode precisar dessas informações para validação, resolução de conflitos, modificação ou exclusão.
- ▶ **Justificativa.** A justificativa fornece a razão pela qual o requisito é necessário, os objetivos das partes interessadas que devem ser atendidos por ele.
- ▶ **Dificuldade.** Esta é uma estimativa do esforço necessário para implementar o requisito. É necessário para o planejamento e estimativa do projeto.
- ▶ **Tipo.** Este atributo indica se o requisito é um requisito funcional ou de qualidade ou uma restrição.

Existem muitas maneiras de armazenar essas informações. Pode estar contido em documentos ou armazenado em uma planilha ou banco de dados, com os requisitos como linhas e seus atributos como colunas. Em configurações ágeis, os requisitos podem ser registrados em cartões de história, onde as rubricas no cartão são os atributos. Conforme discutido no Capítulo 7, as ferramentas de gerenciamento de requisitos devem oferecer funcionalidade para armazenar dados sobre requisitos e também relatar sobre eles.

Os atributos permitem que você forneça informações sobre seus produtos de trabalho e os requisitos neles contidos. A maneira mais simples de fazer isso é produzir um relatório com todos os dados de todas as versões de todos os requisitos. Para qualquer sistema, exceto o mais simples, esse relatório será inútil, pois ninguém será capaz de supervisionar todas as informações porque é extremamente complexo. Portanto, você deve ajustar seus relatórios com base nas necessidades de informações de seu público-alvo. Isso é feito usando *visualizações* [Glin2020]. Uma exibição é uma maneira (muitas vezes predefinida) de filtrar e classificar os dados em seus produtos de trabalho, resultando em um relatório que mostra precisamente o que o público precisa, nem mais nem menos. Uma visão é definida com o propósito explícito de fornecer informações relevantes para um grupo-alvo específico.

Visualizações

Nós discernimos três tipos de pontos de vista:

Tipos de visualizações

- ▶ **Visões seletivas.** Essas exibições fornecem informações sobre uma seleção deliberada dos requisitos em vez de todos os requisitos. Por exemplo, uma visualização apenas das versões mais recentes dos requisitos, ou de todos os requisitos com o estado *validado*, ou dos requisitos com *alta prioridade de stakeholders*; o foco pode estar em um subsistema ou, ao contrário, fornecer uma visão geral abstrata do sistema apenas por meio de seus requisitos de alto nível.
- ▶ **Visões projetivas.** Uma visão projetiva mostra uma seleção de todos os dados (atributos) dos requisitos - por exemplo, apenas a identificação, o número da versão e o nome.
- ▶ **Agregar visualizações.** Em uma visualização agregada, você encontrará resumos, totais ou médias, calculados a partir de um conjunto de requisitos. Um exemplo seria o número total de requisitos por departamento: por exemplo, 4 de Vendas, 5 de Logística.

A Figura 6.3 dá um exemplo desses tipos de visualizações.

		Projective view (only 4 attributes)				
Selective view (only sales dept)	ID	Version	Name	Type	Source	Difficulty (1..5)
	1	2.0	Calculate	Functional	Sales	3
	2	1.0	Response	Quality	Sales	2
	3	1.2	VAT	Constraint	Sales	1
	4	0.2	Foreign VAT	Constraint	Sales	4
		5	Delivery date	Functional	Logistics	2
		6	Track & trace	Functional	Logistics	5
		7	Courier	Functional	Logistics	1
		8	Routing	Functional	Logistics	4
		9	Accessi-bility	Quality	Logistics	3

Aggregating view		Functional	5	Quality	2	Constraint	2

Figura 6.3 Diferentes tipos de visualizações

Na maioria dos casos, uma combinação de visualizações é usada — por exemplo, se você deseja fornecer uma lista com os IDs, números de versão, nomes e tipos (= projetivos) de todos os requisitos para o departamento de Vendas (= seletivo).

6.6 Rastreabilidade

Ao longo deste manual, mencionamos o tópico de rastreabilidade [GoFi1994].

Rastreabilidade

Sem rastreabilidade adequada, a Engenharia de Requisitos dificilmente é viável, pois você não pode fazer o seguinte:

- Fornecer evidências de que um determinado requisito foi satisfeito
- Provar que um requisito foi implementado e por quais meios Mostrar a conformidade do produto com as leis e padrões aplicáveis Procurar produtos de trabalho
- ausentes (por exemplo, descobrir se existem casos de teste para todos os requisitos)
- Analise os efeitos de uma mudança nos requisitos (consulte a Seção 6.7)

Em muitos casos, especialmente para sistemas críticos de segurança, os padrões de processo exigem explicitamente a implementação da rastreabilidade.

Existem três tipos de perguntas que podem ser respondidas com o auxílio da rastreabilidade (veja também a Figura 6.4):

Tipos de rastreabilidade

- **Rastreabilidade reversa:** Qual foi a origem de um determinado requisito? Onde foi encontrado? Quais fontes (partes interessadas, documentos, outros sistemas) foram analisadas durante a elicitação?
- A rastreabilidade reversa é tão conhecida quanto a rastreabilidade de especificação de pré-requisitos.
- **Rastreabilidade futura:** onde esse requisito é usado? Quais entregáveis (módulos codificados, casos de teste, procedimentos, manuais) são baseados nele?
- A rastreabilidade futura é tão conhecida quanto a rastreabilidade de especificação pós-requisitos.
- **Rastreabilidade entre requisitos:** Outros requisitos dependem deste requisito ou vice-versa (por exemplo, requisitos de qualidade relacionados a um requisito funcional)? O requisito é um refinamento de um requisito de nível superior (por exemplo, um épico refinado em várias histórias de usuário, uma história de usuário detalhada com vários critérios de aceitação)? Como eles estão relacionados?

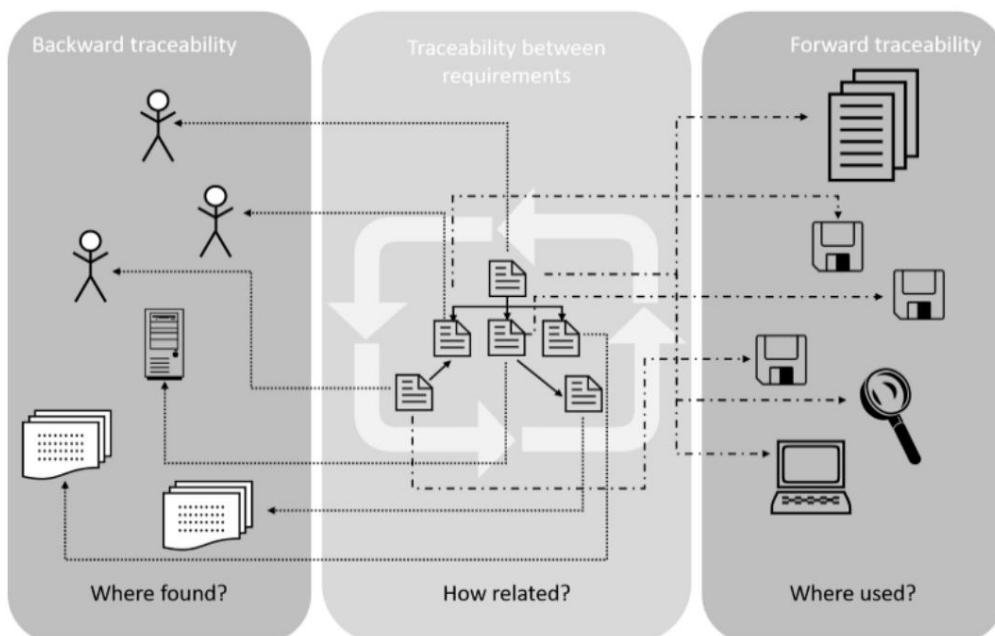


Figura 6.4 Tipos de rastreabilidade

Existem várias maneiras de documentar a rastreabilidade. Frequentemente, isso é feito *implicitamente*, por exemplo, aplicando estruturas de documentos, modelos padrão ou convenções de nomenclatura. Se você identificar todos os seus requisitos com o código *Req-xxx-nnn*, onde *xxx* representa o departamento que solicitou o requisito, todos entenderão que o *Req-sal-012* é um requisito do departamento de Vendas (para rastreabilidade reversa). Se você publicar um documento listando todos os requisitos que serão implementados na versão de 1º de julho, estará fornecendo informações implícitas de rastreabilidade futura. E se você escrever um documento com uma seção dedicada a, por exemplo, cálculos de preços, isso pode ser um exemplo de rastreabilidade entre os requisitos. Outro exemplo poderia ser um modelo de alto nível e uma descrição textual de requisitos detalhados relacionados a ele.

Em projetos mais complexos, a rastreabilidade deve (também) ser documentada *explicitamente*. Para rastreabilidade explícita, você documenta o relacionamento entre os produtos de trabalho com base em sua identificação exclusiva. Isso pode ser feito de várias formas [HuJD2011]:

Documentando o relacionamento entre os produtos de trabalho

- Fazendo uso de atributos específicos como *Source* sugeridos pelo padrão ISO [ISO29148]
- Em documentos, adicionar referências a documentos predecessores, outros produtos de trabalho ou requisitos individuais Desenvolver uma matriz de rastreabilidade em uma planilha
- ou tabela de banco de dados (veja um exemplo na Tabela 6.1 abaixo)
- Na documentação textual, usando hiperlinks no estilo Wiki
- Visualizando relacionamentos de rastreabilidade em um *gráfico de rastreamento* (a Figura 6.4 é uma forma simplificada de tal gráfico)

Em muitos casos, uma ferramenta de gerenciamento de requisitos ou gerenciamento de configuração (consulte o Capítulo 7) fornece funcionalidade para oferecer suporte à rastreabilidade. Gerenciar a rastreabilidade em um projeto substancial pode ser complicado, especialmente se você também tiver que levar em consideração o controle de versão. Nesse caso, um bom ferramental é indispensável.

Tabela 6.1 Exemplo de matriz de rastreabilidade

Fonte	R1	R2	R3	R4	R5	R6	R7
<i>Entrevista Sra. Smith 06/08</i>	x	x			x		
<i>Questionário resumido 12 de maio</i>	x			x		x	x
<i>Relatório de observação de campo 07/03</i>			x	x	x		
<i>Regulamentos da empresa versão 17.a.02</i>			x			x	x
<i>API de documentação do sistema HRM v3.0.2.a</i>	x			x	x		

6.7 Lidando com a Mudança

“Princípio 7: Aceitar mudanças de requisitos, mesmo no final do desenvolvimento. Os processos ágeis aproveitam a mudança para a vantagem competitiva do cliente.” [BeeA2001].

Lidando com a mudança

Os pais fundadores do movimento ágil foram muito claros sobre isso: mudanças de requisitos sempre ocorrerão, goste você ou não. Muitas pessoas não gostam nada de mudanças, porque toda mudança é um risco, uma ameaça à estabilidade do projeto e do sistema.

No entanto, alterar um requisito não é um evento isolado: ele é acionado por mudanças no contexto do sistema, por novos insights dos interessados, pelo comportamento dos concorrentes e assim por diante; uma lei entra em vigor, acrescentando uma nova restrição ao sistema; devido à crescente demanda do mercado, o desempenho do sistema precisa ser melhorado; um sistema concorrente é lançado com alguns recursos *encantadores* que seu cliente também deseja. Uma mudança deve, portanto, ser vista como uma chance de obter um sistema melhor, para fornecer mais valor aos usuários.

Porém, independente da situação, toda mudança também é um risco. Pode introduzir defeitos, levando à falha do sistema. Isso pode atrasar o andamento do projeto. pode demorar

mais esforço e dinheiro do que foi calculado antes. Os usuários podem não gostar e se recusar a trabalhar com ele. Em suma, as coisas podem dar errado e atrapalhar um projeto ou sistema previamente estável. Mas isso não significa que as mudanças sejam ruins e devam ser evitadas; isso significa que todas as alterações devem ser tratadas com cuidado para obter o valor ideal a custos aceitáveis com risco mínimo.

Na literatura sobre gerenciamento de serviços de TI (consulte [Axelos2019]), a *ativação de mudanças* é descrita como uma das principais práticas. Essa prática garante que as mudanças sejam implementadas de forma eficaz, segura e em tempo hábil, a fim de atender às expectativas das partes interessadas. A prática equilibra eficácia, rendimento, conformidade e controle de risco. Ele se concentra em três aspectos:

- Garantir que todos os riscos tenham sido avaliados com precisão
- Autorizando alterações para prosseguir
- Gerenciando a implementação da mudança

Alterar habilitação

A habilitação de mudança implica que uma organização atribui uma autoridade de mudança para decidir sobre as mudanças e define um processo para lidar com elas. Consulte a Figura 6.5 para obter um esboço desse processo. Essas medidas geralmente são ajustadas à abordagem de desenvolvimento e ao ponto no tempo em que ocorre uma mudança.

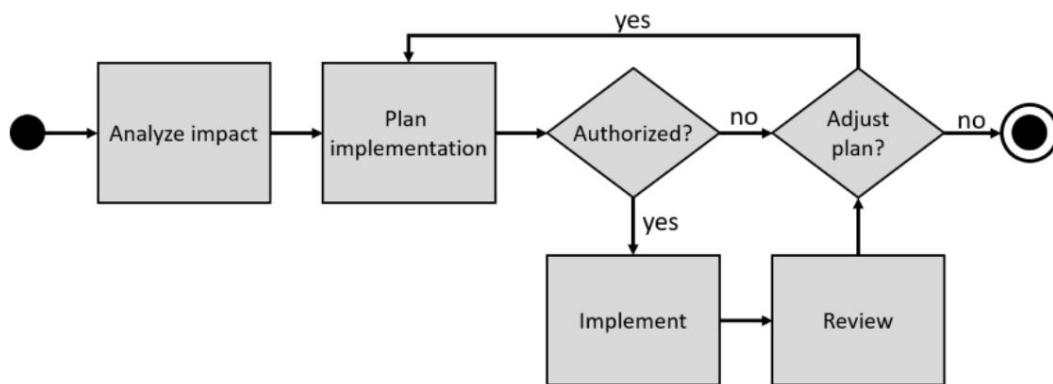


Figura 6.5 Processo de ativação de mudança

Enquanto um requisito estiver em estado de rascunho, o autor tem autoridade para alterá-lo e nenhum processo estrito é seguido.

Assim que um requisito é liberado para uso posterior no projeto, o autor não tem mais liberdade para decidir, pois toda alteração terá impacto em outros produtos de trabalho baseados nesse requisito. Antes de decidir se uma mudança deve ser implementada, uma análise de impacto deve ser realizada para esclarecer os esforços e riscos da mudança.

Alterar autoridade

É aqui que a rastreabilidade é indispensável. Em uma abordagem de desenvolvimento *linear*, a autoridade de mudança geralmente será atribuída ao gerenciamento de projetos, a um comitê de direção ou a um *Conselho de Controle de Mudanças*, e um processo é seguido, com uma decisão formal sobre a mudança e o planejamento de sua implementação. Em uma abordagem de desenvolvimento *iterativo*, a autoridade de mudança geralmente fica com o proprietário do produto, que decide sobre a mudança e adiciona uma mudança aceita ao backlog do produto como um novo item (produto de trabalho). A implementação posterior é tratada como qualquer outro item do backlog do produto.

Uma vez implementado um requisito em um sistema operacional, um processo ainda mais rigoroso deve ser seguido, pois toda mudança passará a influenciar usuários e processos de negócios.

Aqui, muitas vezes é feita uma distinção entre *padrão* (baixo risco, bem compreendido e pré-autorizado, por exemplo, uma alteração na porcentagem do IVA), *normal* (com base em uma solicitação formal de alteração, programada, avaliada e autorizada, por exemplo, uma mudança em um preço

algoritmo de cálculo) e mudanças *de emergência* (a serem implementadas o mais rápido possível, por exemplo, para resolver um incidente - mas isso raramente envolve uma mudança de requisitos). Normalmente, a autoridade de mudança reside em um *Conselho Consultivo de Mudanças* [Math2019]; em uma abordagem iterativa como DevOps, uma mudança pode ser autorizada por um gerente de liberação.

6.8 Priorização

Os próprios requisitos são apenas conceitos na mente das pessoas. Eles agregam valor apenas quando são implementados em um sistema operacional. Essa implementação exige esforço, tempo, dinheiro e atenção. Na maioria dos casos, esses recursos são limitados, o que significa que nem todos os requisitos podem ser implementados, pelo menos não ao mesmo tempo.

Priorização

Isso, por sua vez, significa que as partes interessadas devem decidir quais requisitos devem vir primeiro e quais podem ser implementados posteriormente (ou não). Em outras palavras: priorização [Wieg1999].

A prioridade de um requisito é definida como o nível de importância atribuído a ele de acordo com determinados critérios [Glin2020]. Consequentemente, primeiro você deve determinar quais critérios devem ser usados para avaliar os requisitos antes de poder priorizá-los. No entanto, antes de determinar os critérios de avaliação, você deve saber qual é o objetivo da priorização. Esse objetivo geralmente não é o seu objetivo como Engenheiro de Requisitos, mas o objetivo de certas partes interessadas, portanto, você deve decidir quem são as partes interessadas para essa priorização. E quando você conhece o objetivo deles, geralmente fica claro que nem todos os requisitos terão que ser priorizados, mas apenas um subconjunto definido.

Resumindo o exposto, podemos traçar uma sequência de passos a serem seguidos se quisermos priorizar os requisitos:

Etapas a serem seguidas para priorizar requisitos

► Definir os principais objetivos e restrições para a priorização

O contexto do projeto e do sistema determina em grande parte os motivos da priorização. Se, por exemplo, você priorizar a decisão de quais recursos serão implementados na próxima versão, poderá se concentrar no valor comercial; se o objetivo é selecionar histórias de usuários para a próxima iteração, os pontos da história e as dependências técnicas seriam mais proeminentes. Restrições técnicas ou legais podem limitar as escolhas a serem feitas.

► Defina os critérios de avaliação desejados

Em princípio, os objetivos e restrições ditam os critérios a serem usados.

Os critérios comumente usados são o valor comercial para as partes interessadas, a urgência percebida pelos usuários, o esforço para implementar, os riscos de uso, as dependências lógicas e técnicas, a natureza juridicamente vinculativa de um requisito ou apenas a preferência (inter) subjetiva das partes interessadas relevantes. Às vezes, apenas um único critério é usado, mas uma seleção equilibrada de vários critérios relevantes pode produzir um resultado melhor.

► Definir as *partes interessadas* que devem ser envolvidas

Metas e restrições influenciam quais partes interessadas você deve envolver na priorização, mas, por outro lado, algumas partes interessadas definem essas metas, portanto, você deve estar ciente da interdependência. Por exemplo, ao priorizar o lançamento de um novo sistema, você provavelmente convidaria representantes comerciais e um painel de futuros clientes. Ao priorizar

o backlog do produto para decidir sobre a próxima iteração, a equipe scrum estaria envolvida.

Defina os *requisitos* que devem ser priorizados

É improvável que todo o conjunto de requisitos tenha que ser priorizado. Mais uma vez, isso depende principalmente das metas e restrições para priorizar. Por exemplo, as restrições podem determinar que certos requisitos sejam obrigatórios. Na verdade, só é útil priorizar os requisitos para os quais você tem a opção de incluí-los ou não em uma próxima etapa do processo de desenvolvimento. Isso significa que a fase do projeto também é um fator importante. Em uma fase inicial, você pode incluir versões preliminares na priorização; em uma fase tardia, você frequentemente restringirá a priorização aos requisitos que estão em uma versão estável. Esteja ciente de que os requisitos a serem priorizados devem estar em um nível comparável de abstração, dependendo das metas de priorização. Em uma fase inicial do projeto, por exemplo, você pode priorizar temas ou recursos enquanto prioriza as histórias do usuário no planejamento da iteração.

Selecione a *técnica* de priorização

Uma técnica de priorização é a maneira pela qual as partes interessadas priorizam os requisitos. Conforme descrito abaixo, existem várias técnicas, que diferem em esforço, meticulosidade e nível de detalhe. Aqui também, as metas e restrições definem o cenário, mas o fator mais importante é que as partes interessadas concordem com a técnica que pretendem usar. Caso contrário, eles não aceitarão o resultado e seu esforço de priorização será em vão.

Execute priorização

Quando toda a preparação estiver concluída, a priorização real poderá ser executada. Primeiro, todos os critérios de avaliação definidos devem ser aplicados a todos os requisitos selecionados. Juntamente com as partes interessadas envolvidas, você aplica a técnica de priorização selecionada aos requisitos avaliados. Como resultado, você obtém uma lista priorizada de requisitos. No entanto, pode haver um problema. Diferentes partes interessadas podem ter prioridades diferentes, mesmo que concordem com os critérios avaliados. Nesse caso, você normalmente tem um conflito de requisitos que deve ser resolvido como qualquer outro conflito, conforme descrito na Seção 4.3 sobre resolução de conflitos.

Analizando mais de perto as técnicas de priorização, distinguimos entre duas categorias:

[Categorias de técnicas de priorização](#)

Técnicas ad hoc Com

técnicas ad hoc, os especialistas atribuem prioridades aos requisitos selecionados com base em sua própria experiência. Em princípio, essa priorização é baseada em um único critério, sendo a percepção subjetiva do especialista. Se esse conhecimento for de alto nível e aceitável para as partes interessadas, essa técnica pode ser uma maneira rápida, barata e fácil de obter a priorização. Uma variante seria convidar vários especialistas e calcular algum tipo de prioridade média. Técnicas ad hoc comuns incluem classificação Top-10 e priorização MoSCoW (deve ter, deveria ter, poderia ter, não terá desta vez). A análise de Kano (Seção 4.2.1) também é útil: os insatisfatórios são obrigatórios, os satisfatórios devem

tem, e os deleitadores podem ter ou não ter. Para obter mais informações, consulte, por exemplo, [McIn2016].

► Técnicas analíticas As

técnicas analíticas empregam um processo sistemático para atribuir prioridades. Nessas técnicas, os especialistas atribuem pesos a vários critérios de avaliação (como benefício, custo, risco, tempo de implementação etc.) e, subsequentemente, as prioridades dos requisitos são calculadas como resultados ponderados com base nesses critérios. Essas técnicas exigem mais esforço e tempo, mas têm a vantagem de fornecer uma visão clara dos fatores que determinam as prioridades e do processo pelo qual as prioridades são estabelecidas. Isso pode estimular a aceitação do resultado entre as partes interessadas. No entanto, dois aspectos devem ser mantidos em mente. Primeiro, o resultado é fortemente influenciado pelos fatores de ponderação usados no cálculo do resultado. Portanto, um acordo entre as partes interessadas sobre esses fatores de peso deve ser estabelecido antes da priorização real. Caso contrário, alguns podem tentar alterar os fatores de peso para manipular as prioridades. O segundo aspecto a considerar é que os critérios avaliados são em sua maioria estimativas, não fatos medidos. E as estimativas geralmente estão em uma escala ordinal simples, como baixo, médio, alto. Assim, a qualidade das estimativas é decisiva para a qualidade da priorização resultante.

No entanto, as técnicas analíticas são úteis para fornecer uma priorização claramente fundamentada que seja compreendida e, portanto, aceita pelas partes interessadas envolvidas. Para obter uma explicação detalhada das técnicas analíticas, consulte [Olso2014].

Pode ser tentador aplicar técnicas detalhadas e completas e gastar muito tempo produzindo estimativas perfeitamente precisas em termos de dinheiro, horas, números de vendas esperados etc., e o requisito C de 20.29. Você concluiria então que, evidentemente, C deve ser feito primeiro e A antes de B. No entanto, você provavelmente acabou de introduzir uma pseudo-precisão com esse cálculo e seria melhor concluir que esses três requisitos são igualmente importantes, o que pode ter sido seu pressentimento desde o início. Certifique-se sempre de que o esforço que você gasta na priorização seja justificado pelo valor de uma priorização correta em si. Então, mais uma vez, mantenha os objetivos em mente e lembre-se do Princípio 1: orientação de valor.

6.9 Leitura Adicional

Os livros didáticos de Pohl [Pohl2010], Davis [Davi2005], Hull, Jackson e Dick [HuJD2011], van Lamsweerde [vLam2009] e Wiegers e Beatty [WiBe2013] fornecem uma visão abrangente do gerenciamento de requisitos. Informações adicionais sobre o tópico de gerenciamento de requisitos estão consolidadas no manual CPRE Advanced Level para Gerenciamento de Requisitos de Bühne e Herrmann [BuHe2019].

Cleland-Huang, Gotel e Zisman [CIGZ2012] fornecem um tratamento aprofundado da rastreabilidade.

Olson [Olso2014] e Wiegers [Wieg1999] lidam com técnicas de priorização.

7 Suporte de ferramentas

Um Engenheiro de Requisitos precisa de ferramentas para praticar seu ofício de maneira adequada - assim como um carpinteiro precisa de suas ferramentas, lápis, martelo, serra e furadeira para projetar e construir uma peça de mobiliário. Sem ferramentas, é difícil ou impossível registrar os requisitos, trabalhar juntos nos requisitos e estar no controle dos requisitos.

Este capítulo examina os diferentes tipos de ferramentas de Engenharia de Requisitos (ER) disponíveis e os aspectos que precisam ser levados em consideração para introduzir ferramentas de Engenharia de Requisitos em uma organização.

7.1 Ferramentas na Engenharia de Requisitos

A Engenharia de Requisitos é uma tarefa difícil sem o apoio de ferramentas. As ferramentas são necessárias para dar suporte às tarefas e atividades da Engenharia de Requisitos. As ferramentas existentes se concentram no suporte a tarefas específicas, como documentação de requisitos ou suporte ao processo de RE, e raramente em todas as tarefas e atividades no processo de Engenharia de Requisitos. Portanto, não é surpreendente que o Engenheiro de Requisitos deva ter um conjunto de ferramentas à sua disposição para dar suporte aos vários componentes do processo de Engenharia de Requisitos - assim como o carpinteiro precisa de várias ferramentas (por exemplo, projeto auxiliado por computador (CAD)) para projetar um peça de mobiliário e precisa de ferramentas como serra, raspador e lixa para realizá-la.

Apoio necessário para Requisitos processo de engenharia

As ferramentas são apenas um auxílio ao processo de Engenharia de Requisitos e ao Engenheiro de Requisitos, e tais ferramentas são chamadas de ferramentas CASE (engenharia de software auxiliada por computador). As ferramentas CASE suportam uma tarefa específica no processo de produção de software [Fugg1993].

Nós diferenciamos entre diferentes tipos de ferramentas que suportam os seguintes aspectos da Engenharia de Requisitos:

Diferentes tipos de ferramentas

- Gerenciamento de requisitos As

ferramentas desta categoria possuem as propriedades necessárias para dar suporte às atividades e tópicos descritos no Capítulo 6. Com esses tipos de ferramentas, mais controle pode ser estabelecido sobre o processo de Engenharia de Requisitos. Os requisitos estão sujeitos a mudanças e em um ambiente onde isso acontece com frequência, uma ferramenta com as propriedades relevantes é indispensável. As ferramentas nesta categoria suportam: o

Definição e armazenamento de atributos de requisitos para identificar e coletar dados sobre produtos de trabalho e requisitos conforme descrito na Seção 6.5

o Facilitação e documentação da priorização de requisitos
(Seção 6.8)

o Gerenciamento do ciclo de vida, controle de versão, configurações e linhas de base conforme descrito nas Seções 6.2, 6.3 e 6.4

o Acompanhamento e rastreamento de requisitos, bem como defeitos nos requisitos e produtos de trabalho (Seção 6.6)

o Gestão de mudanças para requisitos; como aprendemos na Seção 6.7, as mudanças são inevitáveis e devem ser gerenciadas com cuidado

Processo de Engenharia de Requisitos

Para dar suporte ao processo de Engenharia de Requisitos, informações são necessárias para permitir que o processo seja ajustado ou melhorado. Este tipo de ferramenta pode:

- o Medir e relatar o processo de Engenharia de Requisitos e fluxo de trabalho
- o Esta informação ajuda a melhorar o processo de Engenharia de Requisitos e reduz o desperdício.
- o Medir e relatar a qualidade do produto e Essas informações ajudam a encontrar defeitos e falhas, que por sua vez podem ser usadas para melhorar a qualidade do produto.

Documentação do conhecimento sobre os requisitos

A quantidade de conhecimento (e requisitos) acumulada em um projeto pode ser enorme. Além disso, uma grande quantidade de conhecimento é construída sobre um produto durante seu ciclo de vida. Todas as informações relevantes devem ser cuidadosamente documentadas para permitir o seguinte:

- o Compartilhamento e criação de um entendimento comum dos requisitos
- o Garantir os requisitos como uma obrigação legal e Uma visão geral e uma visão dos requisitos

Modelagem de requisitos

Como aprendemos na Seção 3.4.1.6, expressar requisitos tanto em diagramas quanto em linguagem natural usa os pontos fortes de ambas as formas de notação. Uma ferramenta capaz de modelar requisitos permite: o Estruturar seus próprios pensamentos; pode ser usado como uma ajuda para pensar o Especificar os requisitos em uma linguagem mais formal do que os requisitos textuais, com todos os benefícios que isso traz

Colaboração na Engenharia de Requisitos

Quando várias pessoas e disciplinas trabalham em um mesmo projeto, uma ferramenta pode apoiar e possibilitar essa colaboração, principalmente no mundo em que vivemos, onde cada vez mais atividades são realizadas localmente (em casa). Este tipo de ferramenta suporta a elicitação, documentação e gerenciamento de requisitos.

Teste e/ou simulação dos requisitos

As ferramentas estão se tornando cada vez mais sofisticadas. Mais e mais opções estão sendo desenvolvidas para testar e/ou simular requisitos com antecedência. Isso permite uma melhor previsão de se os requisitos propostos terão o efeito pretendido.

As ferramentas disponíveis são muitas vezes uma mistura das anteriores. Conforme mencionado anteriormente, diferentes ferramentas podem precisar ser combinadas para suportar adequadamente a Engenharia de Requisitos. Caso sejam utilizadas ferramentas diferentes, é importante atentar para a integração entre elas e a interação com outras aplicações e sistemas para garantir um bom funcionamento.

Às vezes, outros tipos de ferramentas (por exemplo, ferramentas de escritório ou de rastreamento de problemas) são usadas, ou melhor, mal utilizadas, para documentar ou gerenciar requisitos. No entanto, essas ferramentas têm suas limitações e devem ser usadas apenas quando os Engenheiros de Requisitos e as partes interessadas estiverem no controle do processo de RE e os requisitos estiverem alinhados. Caso contrário, este é um grande risco no processo de ER, pois tais ferramentas não suportam nenhuma atividade de gerenciamento de requisitos.

O uso indevido de ferramentas pode comprometer o RE processo

7.2 Apresentando Ferramentas

Selecionar uma ferramenta RE não é diferente de selecionar uma ferramenta para qualquer outra finalidade. Você deve descrever os objetivos, contexto e requisitos antes de selecionar e implementar a(s) ferramenta(s) apropriada(s).

Apresentando ferramentas

As ferramentas são apenas uma ajuda para o processo de Engenharia de Requisitos e para o Engenheiro de Requisitos. Eles não resolvem questões organizacionais ou humanas. Imagine que, junto com seus colegas, você deseja documentar os requisitos de maneira uniforme.

As ferramentas podem suportar isso, por exemplo, com um modelo em uma ferramenta de processamento de texto ou página wiki. Isso não garante que todos os seus colegas adotem esse método de trabalho, nem garante que seus colegas tenham disciplina para registrar e gerenciar seus requisitos dessa maneira. O que pode ajudar é fazer acordos entre si, verificar se os acordos estão sendo cumpridos e poder se comunicar entre si se os acordos não forem cumpridos. Uma ferramenta não vai ajudá-lo com isso. A introdução de uma ferramenta de Engenharia de Requisitos requer responsabilidades e procedimentos claros de Engenharia de Requisitos.

Uma ferramenta pode ajudá-lo a configurar seu processo de Engenharia de Requisitos de forma eficaz e eficiente. As ferramentas geralmente fornecem uma estrutura baseada na experiência de melhores práticas. Essas estruturas podem então ser feitas sob medida para se adequar à situação.

Como aprendemos nos capítulos anteriores, as principais atividades de Engenharia de Requisitos não são processos independentes. A seleção das ferramentas de ER adequadas começa com a definição dos objetivos e/ou problemas que se deseja resolver no processo de ER.

O próximo passo é determinar o contexto do sistema (neste caso, o conjunto de ferramentas).

Considere os aspectos do contexto, ou seja, partes interessadas, processos, eventos, etc., e aplique suas habilidades de Engenharia de Requisitos para especificar os requisitos para as ferramentas de ER.

Pratique o que você prega.

As próximas seções descrevem alguns dos aspectos que devem ser levados em consideração ao introduzir uma (nova) ferramenta de Engenharia de Requisitos em sua organização.

7.2.1 Considerar todos os custos do ciclo de vida além dos custos de licença

Os custos mais óbvios, como custos de compra ou custos de licenciamento, geralmente são contabilizados. Além disso, também devem ser considerados custos menos visíveis, como o uso de recursos na implementação, operação e manutenção da ferramenta.

Custos do ciclo de vida

7.2.2 Considere os Recursos Necessários

Especificar os requisitos e supervisionar o processo de seleção requer os recursos necessários, além dos custos mencionados na seção anterior.

Recursos necessários

Pessoas necessárias para orientar o processo de seleção, engenheiros de requisitos, recursos de hardware e outros recursos não devem ser negligenciados. Depois que a ferramenta é colocada em uso, também podem ser necessários recursos para manutenção e suporte ao usuário.

7.2.3 Evite riscos executando projetos-piloto

A introdução de uma nova ferramenta pode ameaçar o controle sobre a base de requisitos atual. Um caos de requisitos pode surgir porque há uma transição do antigo método de trabalho e/ou ferramentas para o novo método de trabalho e ferramentas. A introdução de uma nova ferramenta durante um projeto existente levará irrevogavelmente a um atraso na entrega dos requisitos e até mesmo no projeto.

Executando projetos-piloto

A introdução de uma nova ferramenta, possivelmente com um método de trabalho diferente, deve ser testada em pequena escala, onde os riscos e impactos permanecem administráveis. Existem várias maneiras de fazer isso:

- Aplicar a ferramenta a um projeto/sistema não crítico
- Use a ferramenta de forma redundante ao lado de um projeto existente
- Aplicar a ferramenta a uma situação/projeto fictício
- Importar/copiar os requisitos de um projeto já concluído

Quando você atingir o ponto em que a ferramenta atende aos objetivos e requisitos definidos, ela poderá ser implementada mais amplamente na organização ou em outros projetos.

7.2.4 Avaliar a ferramenta de acordo com critérios definidos

Selecionar a ferramenta apropriada pode ser uma tarefa difícil. A verificação extensiva de se os objetivos e requisitos foram atendidos é uma abordagem padrão na Engenharia de Requisitos. Uma abordagem sistemática que avalia a ferramenta de diferentes perspectivas também contribui para fazer a escolha certa. As seguintes perspectivas podem ser consideradas:

Avaliando ferramentas sistematicamente

➤ Perspectiva do projeto

Este ponto de vista destaca os aspectos de gerenciamento de projetos. A ferramenta suporta o projeto e as informações necessárias no projeto?

➤ Perspectiva do processo

Essa perspectiva verifica o suporte do processo de Engenharia de Requisitos. A ferramenta suporta suficientemente o processo de ER? Pode ser suficientemente adaptado ao processo de ER e método de trabalho existentes?

➤ Perspectiva do usuário

Essa perspectiva verifica o grau de aplicação por parte dos usuários da ferramenta. Essa é uma visão importante porque se os usuários não estiverem satisfeitos com a ferramenta, o risco da ferramenta não ser aceita aumenta. A ferramenta suporta suficientemente a autorização de usuários e grupos? É suficientemente amigável e intuitivo?

➤ Perspectiva do produto

As funcionalidades oferecidas pela ferramenta são verificadas por esse ângulo. Os requisitos são suficientemente cobertos pela ferramenta? Onde os dados são armazenados? Existe um roteiro com as extensões funcionais para a ferramenta? A ferramenta ainda é suportada pelo fornecedor por enquanto?

➤ Perspectiva do fornecedor

Com essa perspectiva o foco recai sobre o atendimento e a confiabilidade do fornecedor. Onde está localizado o fornecedor? Como é organizado o suporte para esta ferramenta?

➤ Perspectiva econômica

Essa perspectiva analisa o caso de negócios: a ferramenta oferece benefícios suficientes em relação aos custos? Quais são os custos (de gestão) para a compra e manutenção? O que a ferramenta fornece para o processo de RE? É necessário um contrato de manutenção (separado)?

➤ Perspectiva da arquitetura Essa

perspectiva avalia como a ferramenta se encaixa na organização (de TI). A tecnologia aplicada é adequada à organização? A ferramenta pode ser suficientemente vinculada a outros sistemas? A ferramenta se encaixa no cenário de TI e está em conformidade com as restrições arquitetônicas?

7.2.5 Instruir os funcionários sobre o uso da ferramenta

Uma vez selecionada uma ferramenta, os usuários devem se familiarizar com as oportunidades que a ferramenta pode agregar ao processo de Engenharia de Requisitos. Os usuários - isto é, os Engenheiros de Requisitos - devem ser treinados em como usar a ferramenta no processo de Engenharia de Requisitos existente. Se os usuários não forem suficientemente treinados, isso pode significar que nem todos os benefícios da ferramenta são aproveitados. De fato, é possível que a ferramenta seja usada incorretamente, com todas as consequências associadas.

*Instruir os funcionários
sobre o uso das ferramentas*

O processo de Engenharia de Requisitos também pode ser alterado em função da ferramenta selecionada. Aspectos do processo de Engenharia de Requisitos que antes não eram possíveis podem ser viabilizados com uma nova ferramenta: por exemplo, gerenciamento adequado de versões, modelagem de requisitos etc. Isso pode significar que novos procedimentos são acordados, modelos são adaptados ou aplicados, mudanças são feitas para o método de trabalho, e assim por diante. O envolvimento do Engenheiro de Requisitos nesta mudança contribui para o sucesso da aceitação da ferramenta.

7.3 Leitura Adicional

A literatura a seguir pode ser consultada para uma visão geral das ferramentas disponíveis e avaliações de ferramentas. Juan M. Carrillo de Gea et. al. fornecem uma visão abrangente do papel das ferramentas de Engenharia de Requisitos [dGeA2011]. O artigo de Barbara Kitchenham, Stephen Linkman, David Law [KiLL1997] descreve e valida um método para avaliação sistemática de ferramentas. Se você estiver procurando por uma ferramenta de ER, uma lista abrangente de ferramentas para Engenharia de Requisitos é fornecida no site da Volere [Vole2020] ou em [BiHe2020].

8 Referências

- [Alex2005] Ian F. Alexander: Uma Taxonomia das Partes Interessadas – Papéis Humanos na Desenvolvimento de sistema. *Jornal Internacional de Tecnologia e Interação Humana* 2005, 1(1), 23–59.
- [AnPC1994] Annie I. Antón, W. Michael McCracken, Colin Potts: Gol Decomposição e Análise de Cenários em Reengenharia de Processos de Negócios. CAiSE (Conferência sobre Engenharia de Sistemas de Informação Avançada), 1994, 94–104.
- [Armo2004] Philip G. Armour. *As Leis do Processo de Software: Um Novo Modelo para a Produção e Gestão de Software*. Boca Raton, Flórida: CRC Press, 2004.
- [Axelos2019] Axelos: ITIL Foundation: ITIL 4 Edition. Axelos Ltda., 2019.
- [BaBo2014] Stéphane Badreau, Jean-Louis Boulanger: *Engenharia de Requisitos*. Paris: Dunod, 2014 (em francês).
- [BeeA2001] Kent Beck et al.: Princípios por trás do Manifesto Ágil.
<http://agilemanifesto.org/principles.html>, 2001. Última visita em agosto de 2022.
- [BiHe2020] Andreas Birk, Gerald Heller: Lista de Gerenciamento de Requisitos Ferramentas. <https://makingofsoftware.com/resources/list-of-rm-tools/>, 2020, _____
última visita em agosto de 2022.
- [Boeh1981] Barry W. Boehm: *Economia de Engenharia de Software*, Englewood Cliffs, Nova Jersey: Prentice Hall, 1981.
- [BoRJ2005] Grady Booch, James Rumbaugh, Ivar Jacobson: *O Guia do Usuário da Linguagem de Modelagem Unificada*, 2ª edição. Reading, MA: Addison-Wesley, 2005.
- [Bour2009] Lynda Bourne: *Gerenciamento de Relacionamento com Partes Interessadas - Um Modelo de Maturidade para Implementação Organizacional*. Farnham: Gower, 2009.
- [BuHe2019] Stan Bühne, Andrea Herrmann: Requisitos do manual
Gestão de acordo com o Padrão IREB – Educação e Treinamento para o Profissional Certificado IREB para Qualificação de Engenharia de Requisitos Nível Avançado Gerenciamento de Requisitos. Karlsruhe: IREB. <https://www.ireb.org/downloads/#cpire-advanced-level-requirements-management-handbook>, 2019. Última visita em agosto de 2022.
-
- [CaDJ2014] Dante Carrizo, Oscar Dieste, Natalia Juristo: Sistematizando Seleção da Técnica de Elicitação de Requisitos. *Tecnologia da Informação e Software* 2014, 56(6), 644–669.
- [Chen1976] Peter P.-S. Chen: O Modelo Entidade-Relacionamento: Rumo a um Unificado View of Data, *ACM Transactions on Database Systems* 1976, 1(1), 9–36.

[CIGZ2012] Jane Cleland-Huang, Olly Gotel, Andrea Zisman (eds.): Rastreabilidade de Software e Sistemas. Londres: Springer, 2012.

[Cock2001] Alistair Cockburn: Escrevendo casos de uso eficazes. Boston: Addison Wesley, 2001.

[Cohn2004] Mike Cohn: Histórias de usuários aplicadas: para desenvolvimento ágil de software. Boston: Addison-Wesley, 2004.

[Cohn2010] Mike Cohn: Sucesso com Agile: Desenvolvimento de software usando Scrum. Upper Saddle River, NJ: Addison-Wesley, 2010.

[CoWe1998] Reidar Conradi, Bernhard Westfechtel: Modelos de versão para software Gerenciamento de configurações. ACM Computing Surveys 1998, 30(2), 232–282.

[DaTW2012] Marian Daun, Bastian Tenbergen, Thorsten Weyer: Requisitos Ponto de vista. In: K. Pohl, H. Hönniger, R. Achatz, M. Broy: Model-Based Engineering of Embedded Systems. Heidelberg: Springer, 2012.

[Davi1993] Alan M. Davis: Requisitos de Software – Objetos, Funções e Estados. 2ª Edição, Englewood Cliffs, New Jersey: Prentice Hall, 1993.

[Davi1995] Alan M. Davis: 201 Princípios de Desenvolvimento de Software. Nova York: McGraw-Hill, 1995.

[Davi2005] Alan M. Davis: Gerenciamento de requisitos suficiente - Onde O desenvolvimento de software encontra o marketing. Nova York: Dorset House, 2005.

[DeBo2005] Edward De Bono: De Bono's Thinking Course (Revised Edition), Barnes & Noble Books, 2005.

[DeCo2007] Design Council: 11 Lições: Um Estudo do Processo de Design.

<https://www.designcouncil.org.uk/resources/report/11-lessons-management-design-global-brands>, 2007. Última visita em agosto de 2022.

[dGeA2011] Juan M. Carrillo de Gea, Joaquín Nicolás, José L. Fernandez-Alemán, Ambrosio Toval, Christof Ebert, Aurora Vizcaíno: Ferramentas de Engenharia de Requisitos. Software IEEE 2011, 28(4), 86–91.

[DeMa1978] Tom DeMarco: Análise Estruturada e Especificação de Sistema. Novo York: Yourdon Press, 1978.

[DIN66001] DIN 66001:1983-12: Processamento de informações; símbolos gráficos e sua aplicação. Instituto Alemão de Padronização eV, Berlim, 1983 (em alemão).

[Eber2014] Christof Ebert: Engenharia Sistemática de Requisitos, 5ª edição. Heidelberg: dpunkt 2014 (em alemão).

[Fowl1996] Martin Fowler: Padrões de Análise: Modelos de Objetos Reutilizáveis. Reading, MA: Addison-Wesley, 1996.

- [FLCC2016] Xavier Franch, Lidia Lopez, Carlos Cares, Daniel Colomer. (2016). O i* Framework para Modelagem Orientada a Objetivos. Em Domain Specific Conceptual Modeling, Springer, 485-506.
- [Fugg1993] Alfonso Fuggetta: Uma Classificação da Tecnologia CASE. IEEE Computador 1993, 26(12), 25–38.
- [GaWe1989] Donald C. Gause e Gerald M. Weinberg: Explorando Requisitos: Qualidade antes do Projeto. Nova York: Dorset House, 1989.
- [GFPK2010] Tony Gorscheck, Samuel Fricker, Kenneth Palm e Steven A. Kunsman: Um processo de inovação leve para desenvolvimento de produtos com uso intensivo de software. Software IEEE 2010, 27(1), 37–45.
- [GGJZ2000] Carl A. Gunter, Elsa L. Gunter, Michael Jackson, Pamela Zave: A Modelo de Referência para Requisitos e Especificações. IEEE Software 2000, 17(3), 37–43.
- [GHJV1994] Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides: Design Pattern – Elements of Reusable Object-Oriented Software. Reading, Massachusetts: Addison-Wesley, 1994.
- [Gilb1988] Tom Gilb: Princípios de Gerenciamento de Engenharia de Software. Leitura, Mass.: Addison Wesley, 1988.
- [Vidro1999] Friedrich Glasl: Confrontando Conflitos – Um Kit de Primeiros Socorros para Lidar com Conflitos. Stroud, Gloucestershire: Hawthorn Press, 1999.
- [GIFr2015] Martin Glinz e Samuel A. Fricker: Sobre Compreensão Compartilhada em Engenharia de Software: Um Ensaio. Ciência da Computação – Pesquisa e Desenvolvimento 2015, 30(3-4), 363–376.
- [Glin2007] Martin Glinz: Sobre requisitos não funcionais. 15ª Conferência Internacional de Engenharia de Requisitos do IEEE, Delhi, Índia, 2007, 21–26.
- [Glin2008] Martin Glinz: Uma abordagem baseada em risco e orientada para o valor dos requisitos de qualidade. IEEE Software 2008, 25(2), 34–41.
- [Glin2016] Martin Glinz: De quanta engenharia de requisitos precisamos? Softwaretechnik-Tendências 2016, 36(3), 19–21.
- [Glin2019] Martin Glinz: Requirements Engineering I. Course Notes, University of Zurich, 2019. https://www.ifi.uzh.ch/en/rgc/courses/archives/hs19/re_i.html#resources. Última visita em agosto de 2022.
-
- [Glin2020] Martin Glinz: Um Glossário de Terminologia de Engenharia de Requisitos. Versão 2.0. <https://www.ireb.org/downloads/#cpref-glossary>, 2020. Última visita em agosto de 2022.
- [GIWi2007] Martin Glinz e Roel Wieringa: Partes interessadas na Engenharia de Requisitos (Introdução dos Editores Convidados). IEEE Software 2007, 24(2), 18–20.

- [GoFi1994] Orlena Gotel, Anthony Finkelstein: Uma análise do problema de rastreabilidade de requisitos. 1ª Conferência Internacional sobre Engenharia de Requisitos, Colorado Springs, 1994, 94–101.
- [GoRu2003] Rolf Goetz, Chris Rupp: Psicoterapia para requisitos do sistema. 2ª Conferência Internacional IEEE sobre Informática Cognitiva (ICCI'03), Londres, 2003, 75–80.
- [Gott2002] Ellen Gottesdiener: Requisitos por Colaboração: Workshops para Definir Necessidades, Boston: Addison-Wesley Professional, 2002.
- [GreA2017] Eduard C. Groen, Norbert Seyff, Raian Ali, Fabiano Dalpiaz, Joerg Doerr, Emitzá Guzmán, Mahmood Hosseini, Jordi Marco, Marc Oriol, Anna Perini, Melanie Stade: A Multidão na Engenharia de Requisitos - A Paisagem e os Desafios. Software IEEE 2017, 34(2), 44–52.
- [Greg2016] Sarah Gregory: “Depende”: Heurística para “Comum o Suficiente” Requisitos. Palestra principal na REFSQ 2016, Essen, Alemanha, 2016.
- [GRL2020] Linguagem de requisitos orientada a objetivos. Universidade de Toronto, Canadá <https://www.cs.toronto.edu/km/GRL>. Última visita em agosto de 2022.
- [GrSe2005] Paul Grünbacher, Norbert Seyff: Negociação de requisitos. Em um. Aurum, C. Wohlin (eds.): Requisitos de software de engenharia e gerenciamento. Berlim: Springer, 2005, 143-162.
- [Hare1988] David Harel. Sobre formalismos visuais. Communications of the ACM 1988, 31(5), 514–530.
- [HoSch2020] Stefan Hofer, Henning Schwentner: Contação de histórias de domínio - um método de modelagem colaborativa. Disponível em Leanpub, <http://leanpub.com/domainstorytelling>. Última visita em agosto de 2022.
- [HuJD2011] Elizabeth Hull, Ken Jackson, Jeremy Dick: Engenharia de Requisitos. 3ª ed., Berlim: Springer: 2011.
- [Hump2017] Aaron Humphrey: Personas de usuários e perfis de mídia social. Persona Studies 2017, 3(2), 13–20.
- [IEEE830] Prática recomendada IEEE para especificações de requisitos de software. IEEE Std 830-1998, 1998.
- [ISO19650] ISO 19650. Organização e digitalização de informações sobre edifícios e obras de engenharia civil, incluindo Modelagem de informações de construção (BIM) – Gerenciamento de informações usando Modelagem de informações de construção – Parte 1 e 2, 2018.
- [ISO5807] ISO/IEC/IEEE 1985-02: Processamento de informações; Símbolos e convenções de documentação para fluxogramas de dados, programas e sistemas, gráficos de rede de programas e gráficos de recursos do sistema. Organização Internacional de Normalização, Genebra, 1985.
- [ISO25010] ISO/IEC/IEEE 25010:2011: Qualidade de Sistemas e Software Requisitos e Avaliação (SQuaRE) – Sistema e software

modelos de qualidade. Organização Internacional de Normalização, Genebra, 2011.

[ISO29148] ISO/IEC/IEEE 29148: Engenharia de Sistemas e Software – Ciclo de Vida Processos – Engenharia de Requisitos. Organização Internacional de Normalização, Genebra, 2018.

[Jack1995] Michael Jackson: Requisitos e especificações de software: um léxico de prática, princípios e preconceitos. Nova York: ACM Press, 1995.

[Jack1995b] Michael Jackson: O Mundo e a Máquina. 17ª Conferência Internacional sobre Engenharia de Software 1995 (ICSE 1995), 287–292.

[Jaco1992] Ivar Jacobson: Engenharia de software orientada a objetos: uma abordagem orientada a casos de uso. Nova York: ACM Press, 1992.

[JaSB2011] Ivar Jacobson, Ian Spence, Kurt Bittner: Use Case 2.0: The Guide to Succeed with Use Cases. Ivar Jacobson International SA, 2011.

[KiLL1997] Barbara Kitchenham, Stephen Linkman, David Law: DESMET: Uma Metodologia para Avaliação de Métodos e Ferramentas de Engenharia de Software. Computing & Control Engineering Journal 1997, 8(3), 120–126.

[KSTT1984] Noriaki Kano, Nobuhiko Seraku, Fumio Takahashi, Shinichi Tsuji: Qualidade atraente e qualidade obrigatória. Hinshitsu (Qualidade – Jornal da Sociedade Japonesa de Controle de Qualidade) 1984, 14(2), 39-48 (em japonês).

[Laue2002] Søren Lauesen: Requisitos de Software: Estilos e Técnicas. Londres: Addison-Wesley, 2002.

[LaWE2001] Brian Lawrence, Karl Wiegert e Christof Ebert: Os principais riscos da engenharia de requisitos. IEEE Software 2001, 18(6), 62–63.

[LiOg2011] Jeanne Liedtka, Tim Ogilvie: Designing for Growth: A Design Thinking Kit de ferramentas para gerentes. Nova York: Columbia University Press, 2011.

[LiSS1994] Odd I. Lindland, Guttorm Sindre, Arne Sølverg: Compreensão Qualidade em Modelagem Conceitual. IEEE Software 1994, 11(2), 42–49.

[LiSZ1994] Horst Licher, Matthias Schneider-Hufschmidt, Heinz Züllighoven: Prototipagem em Projetos de Software Industrial – Fazendo a ponte entre teoria e prática. IEEE Transactions on Software Engineering 1994, 20(11), 825–832.

[LiQF2010] Soo Ling Lim, Daniele Quercia, Anthony Finkelstein: StakeNet: Usando redes sociais para analisar as partes interessadas em projetos de software de grande escala. 32ª Conferência Internacional sobre Engenharia de Software (ICSE 2010), 2010, 295–304.

[MaGR2004] Neil Maiden, Alexis Gizikis, Suzanne Robertson: Provocando a criatividade: imagine como seriam seus requisitos. IEEE Software 2004, 21(5), 68–75.

[Math2019] Joseph Mathenge: Conselho de Controle de Mudanças vs Conselho Consultivo de Mudanças: Qual é a Diferença? <https://www.bmc.com/blogs/change-control-board-vs-change-advisory-board>, 22 de novembro de 2019. Última visita em agosto de 2022.

[McIn2016] John McIntyre: Modelos MoSCoW ou Kano – Como você prioriza?
<https://www.hotpmo.com/management-models/moscow-kano-priorizar>, 20 de outubro de 2016. Última visita em agosto de 2022.

[MNJR2016] Walid Maalej, Maleknaz I Know, Timo Johann e Guenther Ruhe:
Rumo à engenharia de requisitos baseada em dados. Software IEEE 2016, 33(1), 48–54.

[Moor2014] Christopher W. Moore: O Processo de Mediação – Estratégias Práticas para Resolução de Conflitos, 4ª edição. Hoboken, NJ: John Wiley & Sons, 2014.

[MWHN2009] Alistair Mavin, Philip Wilkinson, Adrian Harwood e Mark Novak:
Abordagem Fácil para a Sintaxe de Requisitos (EARS). 17ª Conferência Internacional de Engenharia de Requisitos do IEEE (RE'09), Atlanta, Geórgia, 2009, 317–322.

[NuKF2003] Bashar Nuseibeh, Jeff Kramer, Anthony Finkelstein: Pontos de vista:
Relacionamentos significativos são difíceis! 25ª Conferência Internacional sobre Engenharia de Software (ICSE'03), Portland, Oregon, 2003, 676–681.

[OleA2018] K. Olsen et al.: Certified Tester, Foundation Level Syllabus - Versão 2018. Conselho Internacional de Qualificações de Teste de Software, 2018.

[Olso2014] David Olson: Priorização de matrizes. <http://www.bawiki.com/wiki/Matrix-Prioritization.html>, 2014. Última visita em agosto de 2022.

[OMG2013] Object Management Group: Business Process Model and Notation (BPMN), versão 2.0.2. Documento OMG, formal/2013-12-09. <https://www.omg.org/spec/BPMN/>. Última visita em agosto de 2022.

[OMG2017] Grupo de gerenciamento de objetos: OMG Unified Modeling Language (OMG UML), versão 2.5.1. Documento OMG, formal/2017-12-05. <https://www.omg.org/spec/UML/>. Última visita em agosto de 2022.

[OMG2018] Grupo de gerenciamento de objetos: OMG Systems Modeling Language (OMG SysML™), versão 1.6. Documento OMG, ptc/2018-12-08. <https://www.omg.org/spec/SysML/>. Última visita em agosto de 2022.

[Osbo1948] Alex F. Osborn: Seu poder criativo: como usar a imaginação. C. Scribner's Sons, 1948. (Acessado como reimpressão digital: Read Books Ltd. (epub eBook), abril de 2013).

[Pich2010] Roman Pichler: Agile Product Management with Scrum – Creating Products that Customers Love, Boston: Addison-Wesley, 2010.

[Pohl2010] Klaus Pohl: Engenharia de Requisitos: Fundamentos, Princípios e Técnicas. Berlim-Heidelberg: Springer, 2010.

[PoRu2015] Klaus Pohl, Chris Rupp: Fundamentos da Engenharia de Requisitos: A Guia de Estudo para o Profissional Certificado para o Exame de Engenharia de Requisitos, (2^a ed.). Rocky Nook, Santa Bárbara, 2015.

[Rein1997] Donald G. Reinertsen: Gerenciando a fábrica de design – um produto Kit de ferramentas do desenvolvedor. A Imprensa Livre, 1997.

[Rein2009] Donald G. Reinertsen: Os Princípios do Fluxo de Desenvolvimento de Produto: Desenvolvimento de Produto Lean de Segunda Geração. Redondo Beach, Ca.: Celeritas Publishing, 2009.

[Ries2011] Eric Ries: A startup enxuta: como os empreendedores de hoje usam a inovação contínua para criar negócios radicalmente bem-sucedidos. Nova York: Crown Business, 2011.

[Robe2001] S. Ian Robertson: Resolução de problemas. Hove, East Sussex: Psychology Press, 2001.

[RoRo2012] Suzanne Robertson e James Robertson: Dominando o Processo de Requisitos: Obtendo os Requisitos Certos. 3^a edição. Boston: Addison-Wesley, 2012.

[RuJB2004] James Rumbaugh, Ivar Jacobson, Grady Booch: Manual de referência da linguagem de modelagem unificada, 2^a edição. Reading, MA: Addison Wesley, 2004.

[Rupp2014] Chris Rupp: Engenharia e gerenciamento de requisitos, 6^a edição. Munique: Hanser, 2014 (em alemão).

[SoSa1998] Ian Sommerville e Pete Sawyer: Engenharia de Requisitos: Um Guia de Boas Práticas. Chichester: John Wiley & Sons, 1997.

[SwBa1982] William Swartout e Robert Balzer: Sobre o inevitável entrelaçamento de Especificação e Implementação. Communications of the ACM 1982, 25(7), 438–440.

[Verd2014] Dave Verduyn: Descobrindo o Modelo Kano, em: Modelo Kano, <https://www.kanomodel.com/discovering-the-kano-model>, 2014. Última visita em agosto de 2022.

[vLam2009] Axel van Lamsweerde: Engenharia de requisitos: de objetivos de sistema a modelos UML a especificações de software. Chichester: John Wiley & Sons, 2009.

[Vole2020] Recursos de requisitos do Volere: <https://www.volere.org>. Última visita em agosto de 2022.

[WiBe2013] Karl Wiegers e Joy Beatty: Requisitos de Software. 3^a edição. Redmond, Wa.: Microsoft Press, 2013.

[Wieg1999] Karl E. Wiegers: Primeiras coisas primeiro: priorizando requisitos. <https://www.processimpact.com/articles/prioritizing.pdf>, 1999. Última visita em agosto de 2022.

[ZoCo2005] Didar Zowghi, Chad Coulin: Elicitação de Requisitos: Uma Pesquisa de Técnicas, Abordagens e Ferramentas. Em A. Aurum, C. Wohlin (eds.) Engenharia e Gerenciamento de Requisitos de Software. Berlim: Springer, 2005, 19-46.