HOPKINS, R.; JENKINS, K. *Eating the IT elephant*: moving from Greenfield development to Brownfield. Boston: IBM Press, 2008.

JEFFRIES, R.; MELNIK, G. "TDD: The art of fearless programming." *IEEE Software*, v. 24, 2007. p. 24-30. doi:10.1109/MS.2007.75.

KILNER, S. "Can agile methods work for software maintenance." 2012. Disponível em: <http://www.vlegaci.com/can-agile-methods-work-for-software-maintenance-part-1/>. Acesso em: 17 fev. 2015.

LARMAN, C.; BASILI, V. R. "Iterative and incremental development: a brief history." *IEEE Computer*, v. 36, n. 6, 2003. p. 47-56. doi:10.1109/MC.2003.1204375.

LEFFINGWELL, D. *Scaling software agility*: best practices for large enterprises. Boston: Addison-Wesley, 2007.

_____. *Agile software requirements*: lean requirements practices for teams, programs and the enterprise. Boston: Addison-Wesley, 2011.

MULDER, M.; VAN VLIET, M. "Case Study: distributed Scrum project for dutch railways." *InfoQ*. 2008. Disponível em: <http://www.infoq.com/articles/dutch-railway-scrum>. Acesso em: 30 mar. 2018.

RUBIN, K. S. *Essential Scrum*. Boston: Addison-Wesley, 2013.

SCHATZ, B.; ABDELSHAFI, I. "Primavera gets agile: a successful transition to agile development." *IEEE Software*, v. 22, n. 3, 2005. p. 36-42. doi:10.1109/MS.2005.74.

SCHWABER, K.; BEEDLE, M. *Agile software development with Scrum*. Englewood Cliffs, NJ: Prentice-Hall, 2001.

STAPLETON, J. *DSDM*: business focused development. 2. ed. Harlow, UK: Pearson Education, 2003.

TAHCHIEV, P.; LEME, F.; MASSOL, V.; GREGORY, G. *JUnit in action*. 2. ed. Greenwich, CT: Manning Publications, 2010.

WEINBERG, G. *The psychology of computer programming*. New York: Van Nostrand, 1971.

WILLIAMS, L.; KESSLER, R. R.; CUNNINGHAM, W.; JEFFRIES, R. "Strengthening the case for pair programming." *IEEE Software*, v. 17, n. 4, 2000. p. 19-25. doi:10.1109/52.854064.

4 Engenharia de requisitos

OBJETIVOS

Os objetivos deste capítulo são introduzir requisitos de software e explicar o processo envolvido na descoberta e na documentação desses requisitos. Ao ler este capítulo, você:

- compreenderá os conceitos de requisitos de usuário e requisitos de sistema e por que eles devem ser escritos de maneiras diferentes;
- compreenderá as diferenças entre requisitos de software funcionais e não funcionais;
- compreenderá as principais atividades da engenharia de requisitos: elicitação, análise e validação, e as relações entre elas;
- compreenderá por que o gerenciamento de requisitos é necessário e como ele apoia outras atividades da engenharia de requisitos.

CONTEÚDO

Os requisitos de um sistema são as descrições dos serviços que o sistema deve prestar e as restrições a sua operação. Esses requisitos refletem as necessidades dos clientes de um sistema que atende a um determinado propósito, como controlar um dispositivo, fazer um pedido ou encontrar informações. O processo de descoberta, análise, documentação e conferência desses serviços e restrições é chamado de engenharia de requisitos (ER).

O termo *requisito* não é utilizado consistentemente na indústria de software. Em alguns casos, um requisito é simplesmente uma declaração abstrata de alto nível de um serviço que um sistema deve oferecer ou de uma restrição a um sistema. No outro extremo, é uma definição formal detalhada de uma função do sistema. Davis (1993) explica por que essas diferenças existem:

Se uma empresa deseja assinar um contrato para um grande projeto de desenvolvimento de software, ela deve definir suas necessidades de uma maneira suficientemente abstrata para que não haja uma solução predefinida. Os requisitos devem ser escritos de modo que vários concorrentes possam disputar o contrato, oferecendo, talvez, maneiras diferentes de satisfazer as necessidades da empresa cliente. Depois de assinado o contrato, o contratado deve escrever uma definição mais detalhada do sistema para o cliente, de modo que ele entenda e valide o que o software fará. Esses dois documentos podem ser reunidos em um documento de requisitos do sistema[1].

Alguns dos problemas que surgem durante o processo de engenharia de requisitos são consequência de não separar claramente os diferentes níveis de descrição. Faço uma distinção entre eles usando os termos *requisitos de usuário* para indicar os requisitos abstratos de alto nível e *requisitos de sistema* para indicar a descrição detalhada do que o sistema deve fazer. Os requisitos de usuário e os requisitos de sistema podem ser definidos da seguinte forma:

1. Requisitos de usuário são declarações, em uma linguagem natural somada a diagramas, dos serviços que se espera que o sistema forneça para os usuários e das limitações sob as quais ele deve operar. Esses requisitos podem variar de declarações amplas das características necessárias do sistema até descrições precisas e detalhadas da sua funcionalidade.
2. Os requisitos de sistema são descrições mais detalhadas das funções, dos serviços e das restrições operacionais do sistema de software. O documento de requisitos de sistema (chamado às vezes de especificação funcional) deve definir exatamente o que deve ser implementado. Pode fazer parte do contrato entre o adquirente do sistema e os desenvolvedores de software.

São necessários diferentes tipos de requisitos para transmitir as informações a respeito de um sistema para diferentes tipos de leitores. A Figura 4.1 ilustra a distinção entre requisitos de usuário e de sistema. Esse exemplo do sistema de informação de pacientes para cuidados com a saúde mental (Mentcare) mostra como um requisito de usuário pode ser ampliado para vários requisitos de sistema. Como se pode ver na figura, o requisito do usuário é bem genérico. Os requisitos de sistema fornecem informações mais específicas sobre os serviços e funções que devem ser implementados.

É necessário escrever os requisitos em diferentes níveis de detalhe, pois diferentes tipos de leitores utilizam esses dados de diferentes maneiras. A Figura 4.2 mostra os tipos de leitores dos requisitos de usuário e de sistema. O primeiro grupo geralmente não está preocupado com o modo como o sistema será implementado, e pode ser composto por gerentes que não estejam interessados nos recursos detalhados do sistema. Por sua vez, o segundo grupo precisa saber com mais precisão o que o sistema fará, seja porque estão interessados em saber como ele apoiará os processos da empresa ou porque estão envolvidos na sua implementação.

1 DAVIS, A. M. *Software requirements*: objects, functions and states. Englewood Cliffs, NJ: Prentice-Hall, 1993.

FIGURA 4.1 Requisitos de usuário e requisitos de sistema.

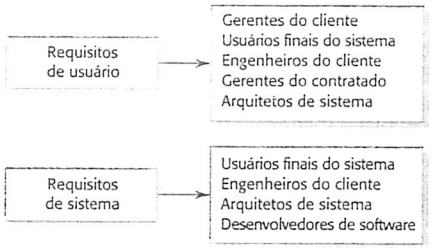
Definição de requisitos de usuário

1. O sistema Mentcare deve gerar relatórios de gestão mensais, mostrando o custo dos medicamentos prescritos por cada clínica naquele mês.

Especificação dos requisitos de sistema

1.1 No último dia útil de cada mês, deve ser gerado um resumo dos medicamentos prescritos, seu custo e a clínica que os prescreveu.
1.2 O sistema deve gerar o relatório para impressão após as 17h30 do último dia útil do mês.
1.3 Deve ser criado um relatório para cada clínica, listando o nome de cada medicamento, a quantidade total de prescrições, a quantidade de doses prescritas e o custo total dos medicamentos prescritos.
1.4 Se os medicamentos estiverem disponíveis em dosagens diferentes (por exemplo, 10 mg, 20 mg etc.) devem ser criados relatórios diferentes para cada dosagem.
1.5 O acesso aos relatórios de medicamentos deve ser restrito aos usuários autorizados, conforme uma lista de controle de acesso produzida pela gestão.

FIGURA 4.2 Leitores dos diferentes tipos de especificação de requisitos.



Os diferentes tipos de leitores de documento exibidos na Figura 4.2 são exemplos de *stakeholders* do sistema[2]. Assim como os usuários, muitas outras pessoas têm algum tipo de interesse no sistema. Os *stakeholders* incluem qualquer um que seja afetado de alguma maneira pelo sistema e, portanto, tenha um interesse legítimo nele. Podem variar de usuários finais de um sistema a gerentes e *stakeholders* externos, como autoridades reguladoras, que certificam a aceitabilidade do sistema. Por exemplo, os *stakeholders* do sistema Mentcare são:

1. pacientes cujas informações estão registradas no sistema e familiares desses pacientes;
2. médicos responsáveis por avaliar e tratar os pacientes;
3. profissionais de enfermagem que coordenam as consultas com os médicos e administram alguns tratamentos;
4. recepcionistas que marcam as consultas dos pacientes;
5. equipe de TI responsável pela instalação e manutenção do sistema;

2 Nota da R.T.: alguns autores traduzem *stakeholders* para "partes interessadas". Porém, o mais comum atualmente é usar o termo em inglês para manter a fidelidade do significado.

6. um gestor de ética médica que deve assegurar que o sistema satisfaz as diretrizes éticas atuais de cuidados com os pacientes;
7. gestores de cuidados com a saúde que obtêm informações gerenciais do sistema;
8. o pessoal de controle do prontuário responsável por garantir que as informações do sistema possam ser mantidas e preservadas e que os procedimentos de manutenção de registros tenham sido adequadamente implementados.

A engenharia de requisitos normalmente é apresentada como o primeiro estágio do processo de engenharia de software. No entanto, pode ser necessário desenvolver algum nível de compreensão dos requisitos de sistema antes de tomar a decisão de adquirir ou desenvolver um sistema. Essa ER inicial estabelece uma visão de alto nível do que o sistema poderia fazer e dos benefícios que poderia proporcionar. Esses pontos podem ser considerados em um estudo de viabilidade, ferramenta usada para avaliar se o sistema é tecnicamente e financeiramente viável. Os resultados desse estudo ajudam a gestão a decidir se deve ou não seguir adiante com a aquisição ou com o desenvolvimento do sistema.

Neste capítulo, apresento uma visão 'tradicional' dos requisitos em vez da visão dos processos ágeis, que discuti no Capítulo 3. Na maioria dos sistemas grandes, ainda é o caso de haver uma fase de engenharia de requisitos claramente identificável antes de começar a implementação do sistema. O resultado é um documento de requisitos, que pode fazer parte do contrato de desenvolvimento do sistema. Naturalmente, são feitas mudanças subsequentes nos requisitos de usuário, que podem ser ampliados para requisitos de sistema mais detalhados. Às vezes, pode-se utilizar uma abordagem ágil para elicitar simultaneamente os requisitos à medida que o sistema é desenvolvido, a fim de acrescentar detalhes e refinar os requisitos de usuário.

Estudos de viabilidade

O estudo de viabilidade é um estudo curto e focalizado que deve ser feito no início do processo de ER. Ele deve responder três perguntas fundamentais:

1. O sistema contribui para os objetivos globais da organização?
2. O sistema pode ser implementado dentro do cronograma e orçamento usando a tecnologia atual?
3. O sistema pode ser integrado com outros sistemas utilizados?

Se a resposta a qualquer uma dessas perguntas for não, provavelmente não se deve prosseguir com o projeto.

4.1 REQUISITOS FUNCIONAIS E NÃO FUNCIONAIS

Os requisitos de sistema de software são classificados frequentemente como funcionais ou não funcionais:

1. *Requisitos funcionais.* São declarações dos serviços que o sistema deve fornecer, do modo como o sistema deve reagir a determinadas entradas e de como deve se comportar em determinadas situações. Em alguns casos, os requisitos funcionais também podem declarar explicitamente o que o sistema não deve fazer.
2. *Requisitos não funcionais.* São restrições sobre os serviços ou funções oferecidas pelo sistema. Eles incluem restrições de tempo, restrições sobre o processo de desenvolvimento e restrições impostas por padrões. Os requisitos não funcionais se aplicam, frequentemente, ao sistema como um todo, em vez de às características individuais ou aos serviços.

Na realidade, a distinção entre os diferentes tipos de requisitos não é tão clara quanto sugerem essas definições simples. Um requisito de usuário relacionado à segurança da informação (*security*), como uma declaração que limita o acesso aos usuários autorizados, pode parecer um requisito não funcional. No entanto, quando desenvolvido em mais detalhes, esse requisito pode gerar outros requisitos claramente funcionais, como a necessidade de incluir no sistema alguns recursos de autenticação do usuário.

Isso mostra que os requisitos não são independentes e que, frequentemente, um requisito gera ou limita outros. Portanto, os requisitos de sistema especificam não apenas os serviços ou características, mas também a funcionalidade necessária para garantir que esses serviços/características sejam entregues corretamente.

4.1 Requisitos funcionais

Os requisitos funcionais de um sistema descrevem o que ele deve fazer e dependem do tipo de software que está sendo desenvolvido, dos usuários esperados para o software e da abordagem geral adotada pela organização ao escrever os requisitos. Quando são apresentados como requisitos de usuário, os requisitos funcionais devem ser escritos de modo compreensível para os usuários e gerentes do sistema. Os requisitos funcionais do sistema expandem os requisitos de usuário e são escritos para os desenvolvedores. Requisitos funcionais devem descrever em detalhes as funções do sistema, suas entradas, saídas e exceções.

Os requisitos funcionais do sistema variam desde os mais gerais, cobrindo o que o sistema deve fazer, até os mais específicos, refletindo os modos de trabalho locais ou os sistemas existentes em uma empresa. A seguir são apresentados exemplos de requisitos funcionais do sistema Mentcare, utilizado para manter informações sobre pacientes recebendo tratamento para problemas de saúde mental:

1. Um usuário deve poder fazer uma busca na lista de consultas de todas as clínicas.
2. O sistema deve gerar, a cada dia e para cada clínica, uma lista de pacientes que devam comparecer às consultas naquele dia.
3. Cada membro da equipe que utiliza o sistema deve ser identificado exclusivamente por seu número de funcionário de oito dígitos.

Esses requisitos de usuário definem funcionalidades específicas que serão incluídas no sistema. Os exemplos mostram que os requisitos funcionais devem ser escritos em diferentes níveis de detalhe (compare os requisitos 1 e 3).

Requisitos de domínio

Requisitos de domínio são derivados do domínio de aplicação do sistema, e não das necessidades específicas de seus usuários. Eles podem ser, em sua essência, novos requisitos funcionais, limitar requisitos funcionais existentes ou estabelecer como determinadas computações devem ser executadas.

O problema com os requisitos de domínio é que os engenheiros de software podem desconhecer as características do domínio no qual o sistema opera, o que significa que eles podem não saber se um requisito de domínio passou despercebido ou se entra em conflito com outros.

Os requisitos funcionais, como o nome sugere, têm focado tradicionalmente no que o sistema deve fazer. No entanto, se uma organização decidir que um sistema de prateleira pode satisfazer suas necessidades, então há muito pouco sentido em desenvolver uma especificação funcional detalhada. Nesses casos, o foco deve ser o desenvolvimento de requisitos de informação que especifiquem as informações necessárias para as pessoas fazerem seu trabalho. Os requisitos de informação especificam as informações necessárias e como elas devem ser fornecidas e organizadas. Portanto, um requisito de informação do sistema Mentcare poderia especificar quais informações devem ser incluídas na lista de pacientes que devem comparecer às consultas do dia.

A imprecisão na especificação de requisitos pode levar a conflitos entre clientes e desenvolvedores de software. É normal que um desenvolvedor de sistemas interprete um requisito ambíguo de uma forma que simplifique a sua implementação. Muitas vezes, porém, não é isso o que o cliente quer. Novos requisitos devem ser estabelecidos e mudanças devem ser feitas, o que resulta em atraso na entrega do sistema e aumento dos custos.

Por exemplo, o primeiro requisito do sistema Mentcare, mencionado anteriormente, afirma que um usuário deve ser capaz de fazer uma busca nas listas de consultas de todas as clínicas. O que justifica esse requisito é que os pacientes com transtornos de saúde mental às vezes se confundem. Eles podem ter uma consulta em uma clínica, mas acabar indo para outra. Se tiverem uma consulta marcada, serão registrados como atendidos, independentemente da clínica.

Um membro da equipe médica, ao especificar um requisito de busca, pode esperar que 'pesquisar' signifique que, dado o nome de um paciente, o sistema procure por ele em todas as consultas de todas as clínicas. No entanto, isso não está explícito. Os desenvolvedores de sistemas podem interpretar o requisito do modo mais fácil de implementar. Sua função de busca pode exigir que o usuário escolha uma clínica e depois faça a pesquisa dos pacientes que compareceram a ela. Isso envolve mais informações fornecidas pelo usuário e leva mais tempo para completar a busca.

Em condições ideais, a especificação dos requisitos funcionais de um sistema deve ser completa – todos os serviços e informações requisitados pelo usuário devem estar definidos – e coerente – os requisitos não devem ser contraditórios.

Na prática, só é possível alcançar a coerência e a completude dos requisitos em sistemas de software muito pequenos, e uma das razões para isso é que é mais fácil cometer erros e omissões quando escrevemos especificações de sistemas grandes

e complexos. Além disso, sistemas grandes possuem muitos *stakeholders*, com diferentes formações e expectativas, e que tendem a ter necessidades diferentes — e, muitas vezes, inconsistentes. Essas inconsistências podem não ser óbvias quando os requisitos são especificados em um primeiro momento, e os requisitos inconsistentes podem ser descobertos somente após uma análise mais profunda ou durante o desenvolvimento do sistema.

4.1.2 Requisitos não funcionais

Os requisitos não funcionais, como o nome sugere, são aqueles que não possuem relação direta com os serviços específicos fornecidos pelo sistema aos seus usuários. Esses requisitos não funcionais normalmente especificam ou restringem as características do sistema como um todo. Eles podem estar relacionados a propriedades emergentes do sistema, como confiabilidade, tempo de resposta e uso da memória. Por outro lado, podem definir restrições à implementação do sistema, como a capacidade dos dispositivos de E/S ou as representações dos dados utilizados nas interfaces com outros sistemas.

Os requisitos não funcionais frequentemente são mais críticos do que os requisitos funcionais individuais. Os usuários do sistema normalmente encontram maneiras de contornar uma função do sistema que não satisfaça suas necessidades. No entanto, descumprir um requisito não funcional pode significar a inutilização total do sistema. Por exemplo, se um sistema de aeronave não satisfizer seus requisitos de confiabilidade, este não será certificado como seguro para operação; se um sistema de controle embarcado não cumprir seus requisitos de desempenho, as funções de controle não vão funcionar corretamente.

Embora muitas vezes seja possível identificar quais componentes do sistema implementam requisitos funcionais específicos (por exemplo, pode haver componentes de formatação que implementem requisitos de relatório), isso é mais difícil com os requisitos não funcionais. Sua implementação pode estar espalhada por todo o sistema por duas razões:

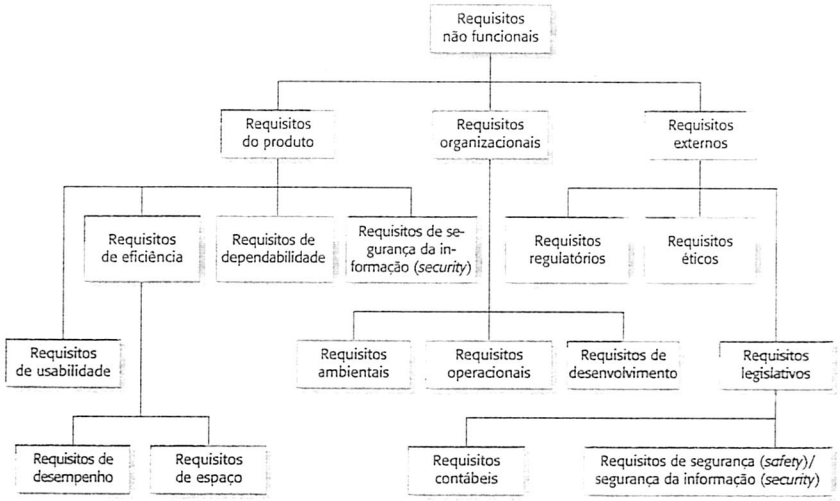
1. Os requisitos não funcionais podem afetar a arquitetura geral de um sistema em vez de seus componentes individuais. Por exemplo, para garantir que sejam cumpridos os requisitos de desempenho em um sistema embarcado, pode ser necessário organizá-lo a fim de minimizar a comunicação entre seus componentes.
2. Um requisito não funcional individual, como um requisito de segurança da informação (*security*), pode gerar vários requisitos funcionais relacionados que definem novos serviços do sistema que se fazem necessários caso o requisito não funcional seja implementado. Além disso, também pode gerar requisitos que restringem outros requisitos existentes; por exemplo, pode limitar o acesso à informação no sistema.

Os requisitos não funcionais surgem das necessidades dos usuários, que se devem a restrições orçamentárias, políticas organizacionais, necessidade de interoperabilidade com outros sistemas de software ou hardware, ou fatores externos, como normas de segurança (*safety*) ou legislação relativa à privacidade. A Figura 4.3 mostra uma classificação dos requisitos não funcionais. É possível ver nesse diagrama que esses

requisitos podem ser provenientes das características exigidas do software (requisitos do produto), da organização que o desenvolve (requisitos organizacionais) ou de fontes externas:

1. *Requisitos do produto.* Esses requisitos especificam ou restringem o comportamento do software durante a execução. Os exemplos incluem requisitos de desempenho, relativos à rapidez com que o sistema deve executar e de quanta memória ele precisa; requisitos de confiabilidade, que estabelecem a taxa de falha aceitável; requisitos de segurança da informação (*security*); e requisitos de usabilidade.
2. *Requisitos organizacionais.* São requisitos de sistema amplos, derivados das políticas e procedimentos nas organizações do cliente e do desenvolvedor. Os exemplos incluem requisitos de processos operacionais, que definem como o sistema será utilizado; requisitos de processos de desenvolvimento, que especificam a linguagem de programação, o ambiente de desenvolvimento ou os padrões de processo a serem utilizados; e os requisitos ambientais, que especificam o ambiente operacional do sistema.
3. *Requisitos externos.* Esse título abrangente cobre todos os requisitos derivados de fatores externos ao sistema e seu processo de desenvolvimento. Podem incluir requisitos regulatórios, que estabelecem o que deve ser feito para o sistema ser aprovado por uma entidade reguladora, como uma autoridade de segurança nuclear; requisitos legislativos, que devem ser seguidos para garantir que o sistema opere dentro da lei; e requisitos éticos, que garantem que o sistema será aceitável para os usuários e para o público em geral.

FIGURA 4.3 Tipos de requisitos não funcionais.



A Figura 4.4 mostra exemplos de requisitos do produto, organizacionais e externos que poderiam estar incluídos na especificação do sistema Mentcare. O requisito do produto é o requisito de disponibilidade que define quando o sistema deve estar disponível e o tempo de parada permitido por dia. Ele nada diz sobre a funcionalidade do sistema Mentcare e identifica claramente uma restrição a ser considerada pelos projetistas do sistema.

FIGURA 4.4 Exemplos de possíveis requisitos não funcionais do sistema Mentcare.

Requisito do produto

O sistema Mentcare deve ficar disponível para todas as clínicas durante o expediente normal (segunda-sexta, 8h30-17h30).

O tempo que o sistema pode permanecer fora do ar no expediente normal não deve ultrapassar 5 segundos em qualquer dia.

Requisito organizacional

Os usuários do sistema Mentcare devem se identificar usando o cartão de identificação de autoridade de saúde.

Requisito externo

O sistema deve implementar providências para a privacidade do paciente, conforme estabelecido em HStan-03-2006-priv.

O requisito organizacional especifica a forma de autenticação dos usuários. A autoridade de saúde que opera o sistema está aplicando em todo o software um procedimento de autenticação padrão que, em vez de ser feito através de um *login*, passa a ser feito por uma leitora que reconhece o cartão de identificação dos usuários. O requisito externo deriva da necessidade de o sistema obedecer a legislação relativa à privacidade. Esta é, obviamente, uma questão muito importante nos sistemas de saúde, e o requisito especifica que o sistema deve ser desenvolvido de acordo com um padrão nacional de privacidade.

Um problema comum com os requisitos não funcionais é que os *stakeholders* propõem esses requisitos na forma de metas gerais, como a facilidade de uso, a capacidade do sistema para se recuperar de uma falha ou a resposta rápida do usuário. As metas estabelecem boas intenções, mas causam problemas para os desenvolvedores do sistema, uma vez que abrem espaço para interpretação e subsequente conflito após o sistema ser entregue. Por exemplo, a meta do sistema a seguir é um exemplo típico de como os requisitos de usabilidade seriam solicitados por um gestor:

O sistema deve ser fácil de usar pela equipe médica e ser organizado de tal modo que os erros de usuário sejam minimizados.

Reescrevi isso para mostrar como a meta poderia ser expressa como um requisito não funcional 'testável'. É impossível verificar de forma imparcial a meta do sistema, mas na descrição a seguir é possível, ao menos, incluir a instrumentação de software para contar os erros cometidos pelos usuários quando estiverem realizando um teste.

A equipe médica deve ser capaz de utilizar todas as funções do sistema após duas horas de treinamento. Após essa etapa, a quantidade média de erros cometidos pelos usuários experientes não deve ultrapassar dois erros por hora de uso do sistema.

Sempre que possível, os requisitos não funcionais devem ser escritos de forma quantitativa para que possam ser testados objetivamente. A Figura 4.5 exibe as métricas para especificar as propriedades não funcionais do sistema. É possível mensurar essas características quando o sistema estiver sendo testado, para conferir se ele cumpriu ou não seus requisitos não funcionais.

FIGURA 4.5 Métricas para especificar requisitos não funcionais.

Propriedade	Métrica
Velocidade	Transações processadas/segundo Tempo de resposta do usuário/evento Tempo de atualização da tela
Tamanho	Megabytes/número de chips de ROM
Facilidade de uso	Tempo de treinamento Número de quadros de ajuda
Confiabilidade	Tempo médio até a falha Probabilidade de indisponibilidade Taxa de ocorrência de falhas Disponibilidade
Robustez	Tempo para reiniciar após a falha Porcentagem de eventos causando falhas Probabilidade de corromper dados em uma falha
Portabilidade	Porcentagem de declarações dependentes do sistema-alvo Número de sistemas-alvo

Na prática, os clientes de um sistema costumam achar difícil traduzir suas metas para requisitos mensuráveis. Para algumas metas, como a manutenibilidade, não há métricas simples que possam ser utilizadas. Em outros casos, quando é possível fazer uma especificação quantitativa, os clientes podem não conseguir relacionar suas necessidades com essas especificações. Eles não entendem, por exemplo, o que algum número definindo confiabilidade significa em termos de experiência cotidiana com sistemas de computador. Além disso, o custo de verificar objetivamente os requisitos não funcionais mensuráveis pode ser muito alto e os clientes que pagam pelo sistema podem achar que esses valores não são justificáveis.

Frequentemente, os requisitos não funcionais entram em conflito e interagem com outros funcionais ou não funcionais. Por exemplo, o requisito de identificação na Figura 4.4 requer que uma leitora de cartão seja instalada em cada computador conectado ao sistema. No entanto, pode haver outro requisito que exija acesso móvel ao sistema, por meio de tablets e smartphones dos médicos e profissionais de enfermagem. Esses dispositivos normalmente não são equipados com leitoras de cartão; portanto, nessas circunstâncias, pode ser necessário o suporte para algum método de identificação alternativo.

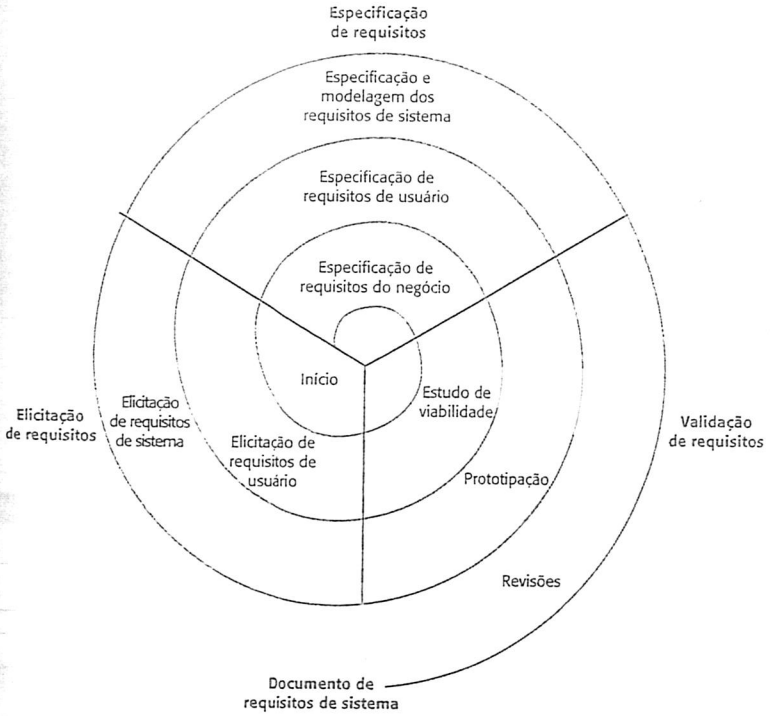
É difícil separar os requisitos funcionais dos não funcionais no documento de requisitos. Se os não funcionais forem declarados separadamente dos funcionais, a relação entre eles pode ser difícil de compreender. Entretanto, em condições ideais, deve-se destacar os requisitos claramente relacionados às propriedades emergentes do sistema, como desempenho ou confiabilidade. É possível fazer isso colocando-os em uma seção separada do documento de requisitos ou distinguindo-os, de alguma forma, dos demais requisitos de sistema.

Os requisitos não funcionais, como confiabilidade, segurança e confidencialidade, são particularmente importantes para os sistemas críticos. Na Parte 2, os requisitos de dependabilidade serão abordados, bem como as maneiras de especificar confiabilidade, segurança (*safety*) e segurança da informação (*security*) na forma de requisitos.

4.2 PROCESSOS DE ENGENHARIA DE REQUISITOS

Conforme discuti no Capítulo 2, a engenharia de requisitos envolve três atividades fundamentais: a descoberta dos requisitos por meio da interação com *stakeholders* (elicitação e análise); a conversão desses requisitos em uma forma padrão (especificação); e a averiguação de que os requisitos realmente definem o sistema que o cliente quer (validação). Mostrei esses processos sequenciais na Figura 2.4. Entretanto, na prática, a engenharia de requisitos é um processo iterativo, no qual as atividades são intercaladas, como mostra a Figura 4.6.

FIGURA 4.6 Uma visão em espiral do processo de engenharia de requisitos.



As atividades são organizadas como um processo iterativo em torno de uma espiral, e o resultado do processo de ER é um documento de requisitos de sistema. A quantidade de tempo e esforço dedicados a cada atividade em uma iteração depende do estágio do processo geral, do tipo de sistema a ser desenvolvido e do orçamento disponível.

No início do processo, a maior parte do esforço é dedicada à compreensão do negócio em alto nível e dos requisitos não funcionais, além dos requisitos de usuário do sistema. Em uma etapa mais avançada do processo — nos anéis mais externos da espiral —, mais esforço será dedicado à elicitação e à compreensão dos requisitos não funcionais e dos requisitos de sistema mais detalhados.

Esse modelo em espiral acomoda abordagens para o desenvolvimento nas quais os requisitos são desenvolvidos em diferentes níveis de detalhe. O número de iterações em torno da espiral pode variar, de modo que ela pode ser encerrada após alguns ou todos os requisitos de usuário terem sido elicitados. O desenvolvimento ágil pode ser utilizado, em vez da prototipação, para que os requisitos e a implementação do sistema sejam desenvolvidos em conjunto.

Em praticamente todos os sistemas, os requisitos mudam. As pessoas envolvidas desenvolvem uma compreensão melhor do que elas querem que o software faça; a organização que está adquirindo o sistema muda; e são feitas modificações no hardware, no software e no ambiente organizacional do sistema. As mudanças devem ser gerenciadas para entender tanto o impacto em outros requisitos quanto as implicações no sistema e o custo de realizá-las. Discutirei esse processo de gerenciamento dos requisitos na Seção 4.6.

4.3 ELICITAÇÃO DE REQUISITOS

Os objetivos do processo de elicitação de requisitos são compreender o trabalho que os *stakeholders* realizam e entender como usariam um novo sistema para apoiar o trabalho deles. Durante a elicitação de requisitos, os engenheiros de software trabalham com os *stakeholders* para saber mais sobre o domínio da aplicação, as atividades envolvidas no trabalho, os serviços e as características do sistema que eles querem, o desempenho desejado para o sistema, as limitações de hardware etc.

Elicitar e compreender os requisitos dos *stakeholders* no sistema é um processo difícil por várias razões:

1. Muitas vezes os *stakeholders* não sabem o que querem de um sistema de computador, exceto em aspectos mais gerais; eles podem achar difícil articular o que querem que o sistema faça; podem fazer exigências irreais porque não sabem o que é viável ou não.
2. Em um sistema, é natural que os *stakeholders* expressem os requisitos em seus próprios termos e com conhecimento implícito de seu próprio trabalho. Os engenheiros de requisitos, sem experiência no domínio do cliente, podem não entender tais requisitos.
3. Diferentes *stakeholders*, com requisitos distintos, podem expressá-los de maneiras variadas. Os engenheiros de requisitos têm de descobrir todas as possíveis fontes de requisitos, além dos pontos de convergência e de conflito.
4. Fatores políticos podem influenciar os requisitos de um sistema. Os gerentes podem exigir requisitos de sistema específicos, o que lhes permite aumentar sua influência na organização.
5. O ambiente econômico e de negócios no qual a análise ocorre é dinâmico. Inevitavelmente, ele muda durante o processo de análise. A importância de determinados requisitos pode mudar. Novos requisitos podem surgir de novos *stakeholders* que não foram consultados originalmente.

Um modelo do processo de elicitação e análise é exibido na Figura 4.7. Cada organização terá sua própria versão ou instanciação desse modelo geral, dependendo de fatores locais como a experiência da equipe, o tipo de sistema sendo desenvolvido e os padrões utilizados.

FIGURA 4.7 O processo de elicitação e análise de requisitos.

As atividades de processo são:

1. *Descoberta e compreensão dos requisitos.* Esse é o processo de interagir com os *stakeholders* do sistema para descobrir seus requisitos. Os requisitos de domínio dos *stakeholders* e documentação também são descobertos durante essa atividade.
2. *Classificação e organização dos requisitos.* Essa atividade pega o conjunto não estruturado de requisitos, agrupa os requisitos relacionados e os organiza em grupos coerentes.
3. *Priorização e negociação dos requisitos.* Inevitavelmente, quando estão envolvidos vários *stakeholders*, os requisitos entrarão em conflito. Essa atividade está relacionada com a priorização dos requisitos e com a descoberta e negociação para resolução de conflitos. Normalmente, os *stakeholders* devem se reunir para resolver as diferenças e chegar a um acordo sobre os requisitos.
4. *Documentação dos requisitos.* Os requisitos são documentados e servem de entrada para a próxima volta da espiral. Um rascunho inicial pode ser produzido nesse estágio ou os requisitos podem simplesmente ser mantidos de modo informal em lousas, *wikis* ou outros espaços compartilhados.

A Figura 4.7 mostra que elicitação e análise de requisitos são processos iterativos, com *feedback* contínuo de cada atividade para as demais. O ciclo do processo começa com a descoberta dos requisitos e termina com sua documentação. A compreensão que o analista tem dos requisitos aumenta a cada rodada do ciclo, que termina quando o documento de requisitos for produzido.

Para simplificar a análise dos requisitos, é útil organizar e agrupar as informações dos *stakeholders*. Uma maneira de fazer isso é considerar cada grupo de *stakeholders* como um ponto de vista e coletar todos os requisitos desse grupo a partir de um determinado ponto de vista. É possível incluir pontos de vista para representar requisitos de domínio e restrições dos outros sistemas. Alternativamente, é possível

usar um modelo da arquitetura do sistema para identificar subsistemas e associar requisitos a cada um deles.

Pontos de vista

Ponto de vista é uma maneira de coletar e organizar um conjunto de requisitos de um grupo de *stakeholders* que têm algo em comum. Portanto, cada ponto de vista inclui um conjunto de requisitos de sistema. Os pontos de vista poderiam vir de usuários finais, gerentes ou outros, e ajudam a identificar as pessoas que podem fornecer informações sobre seus requisitos e estruturá-los para análise.

Inevitavelmente, diferentes *stakeholders* têm opiniões diferentes sobre a importância e a prioridade dos requisitos; às vezes, elas podem entrar em conflito. Se alguns *stakeholders* acharem que suas opiniões não foram consideradas adequadamente, então podem tentar minar deliberadamente o processo de ER. Portanto, é importante organizar reuniões regulares com os *stakeholders*, dando-lhes a oportunidade de expressar suas preocupações e chegar a um acordo quanto aos requisitos.

No estágio de documentação, é importante utilizar uma linguagem simples e diagramas para descrever os requisitos. Isso possibilita que os *stakeholders* os entendam e os comentem. Para facilitar a troca de informações, é melhor utilizar um documento compartilhado (por exemplo, Google Docs ou Office 365) ou um *wiki* que seja acessível a todos os *stakeholders*.

4.3.1 Técnicas de elicitação de requisitos

A elicitação dos requisitos envolve encontros com *stakeholders* de diferentes tipos para descobrir informações sobre o sistema proposto. É possível complementá-las com o conhecimento existente dos sistemas e seu uso, além de informações provenientes de documentos de vários tipos. Há que se investir tempo compreendendo como as pessoas trabalham, o que elas produzem, como usam outros sistemas e como podem precisar mudar para acomodar um novo.

Existem duas abordagens fundamentais para a elicitação de requisitos:

1. entrevistas, em que há uma conversa com as pessoas a respeito do que elas fazem;
2. observação ou etnografia, em que se observa as pessoas executando seu trabalho para ver quais artefatos elas usam, como usam etc.

Deve-se empregar uma mistura de entrevista e observação para coletar informações e, a partir disso, derivar os requisitos que mais tarde embasarão outras discussões.

4.3.1.1 Entrevistas

As entrevistas formais ou informais com os *stakeholders* do sistema fazem parte da maioria dos processos de engenharia de requisitos. Nelas, a equipe de engenharia de requisitos coloca questões para os *stakeholders* sobre o sistema que utilizam no momento e o sistema a ser desenvolvido. Os requisitos são derivados das respostas. As entrevistas podem ser de dois tipos:

1. entrevistas fechadas, nas quais os *stakeholders* respondem a um conjunto predefinido de perguntas;
2. entrevistas abertas, nas quais não há uma programação predefinida. A equipe de engenharia de requisitos explora uma gama de questões com os *stakeholders* do sistema e, a partir daí, desenvolve uma melhor compreensão de suas necessidades.

Na prática, as entrevistas com os *stakeholders* normalmente são uma mistura dos dois tipos. É possível obter a resposta a determinadas perguntas, mas, normalmente, elas levam a outras questões discutidas de uma maneira menos estruturada. As discussões totalmente abertas raramente funcionam bem. Em geral, deve-se fazer algumas perguntas para começar e manter a entrevista focada no sistema a ser desenvolvido.

As entrevistas são boas para obter uma compreensão global do que os *stakeholders* fazem, de como interagiriam com o novo sistema e das dificuldades que enfrentam nos sistemas atuais. As pessoas gostam de falar sobre o trabalho delas e, por isso, normalmente ficam felizes em participar de entrevistas. No entanto, a não ser que haja um protótipo do sistema para demonstrar, não se deve esperar que os *stakeholders* sugiram requisitos específicos e detalhados. Todo mundo acha difícil imaginar como um sistema poderia se parecer, portanto é preciso analisar as informações coletadas e gerar os requisitos a partir disso.

Pode ser difícil obter conhecimento de uma certa área por meio de entrevistas, por duas razões:

1. Todos os especialistas em aplicações usam jargões específicos de sua área de trabalho. É impossível discutir os requisitos sem usar esse tipo de terminologia. Normalmente, usam palavras de maneira precisa e sutil, que os engenheiros podem entender erroneamente.
2. Certos conhecimentos da área podem ser tão familiares aos *stakeholders* que eles acham difícil explicá-los, ou podem ser tão básicos que eles pensam que não vale a pena mencioná-los. Por exemplo, para um bibliotecário, não é preciso dizer que todas as aquisições são catalogadas antes de serem acrescentadas à biblioteca. No entanto, isso pode não ser óbvio para o entrevistador e, por esse motivo, sequer ser levado em conta entre os requisitos.

As entrevistas não são uma técnica eficaz para elicitar conhecimento a respeito dos requisitos e das restrições organizacionais porque existem relações de poder sutis entre as pessoas em uma empresa. As estruturas organizacionais divulgadas raramente correspondem à realidade da tomada de decisão em uma empresa, mas os entrevistados podem não querer revelar para um estranho a estrutura real em vez da teórica. Em geral, a maioria das pessoas reluta em discutir questões políticas e organizacionais que possam afetar os requisitos.

Para uma entrevista eficaz, duas coisas devem ser levadas em conta:

1. Ter a mente aberta, evitar ideias preconcebidas a respeito dos requisitos e ter disposição para ouvir os *stakeholders*. Se ele tiver propostas de requisitos surpreendentes, então é necessário estar disposto a mudar de ideia a respeito do sistema.
2. Incentivar o entrevistado a manter a conversa fazendo uma pergunta que sirva como trampolim ou uma proposta de requisitos; ou então trabalhando juntos em um sistema protótipo. Provavelmente, falar para as pessoas "diga-me o que você quer" não vai resultar em informações úteis, pois é muito mais fácil falar em um contexto definido do que em termos gerais.

As informações das entrevistas são utilizadas junto com outras que dizem respeito ao sistema, como a documentação que descreve os processos do negócio ou dos sistemas existentes, as observações do usuário e a experiência do desenvolvedor. Às vezes, além das informações nos documentos do sistema, as informações da entrevista podem ser a única fonte de informação sobre os requisitos de sistema. Entretanto, a entrevista em si está sujeita à perda de informações essenciais e, portanto, deve ser utilizada em conjunto com outras técnicas de elicitação de requisitos.

4.3.1.2 Etnografia

Os sistemas de software não existem isoladamente. Eles são utilizados em um ambiente social e organizacional e seus requisitos podem ser gerados ou restringidos por ele. Uma razão pela qual muitos sistemas de software são entregues, mas jamais utilizados, é que seus requisitos não levam em conta como esses fatores sociais e organizacionais afetam sua operação prática. Portanto, é muito importante que, durante o processo de engenharia de requisitos, se tente entender os problemas que afetam o uso do sistema.

A etnografia é uma técnica de observação que pode ser utilizada para entender os processos operacionais e para ajudar a derivar os requisitos do software que apoia esses processos. Um analista deve ficar imerso no ambiente de trabalho em que o sistema será utilizado com o objetivo de observar o dia a dia e tomar nota das tarefas reais nas quais os participantes estão envolvidos. A vantagem da etnografia é que ela ajuda a descobrir requisitos implícitos do sistema, os quais refletem o verdadeiro modo de trabalho das pessoas, em vez dos processos formais definidos pela organização.

Frequentemente, as pessoas acham difícil articular detalhes do seu trabalho porque é tão natural para elas que não precisam mais pensar a respeito dele. Elas entendem seu próprio trabalho, mas não a relação que ele possui com outros trabalhos na organização. Fatores sociais e organizacionais que afetam o trabalho, mas que não são óbvios para os indivíduos, podem ficar claros somente quando forem notados por um observador imparcial. Por exemplo, um grupo de trabalho pode se auto-organizar para que os membros conheçam o trabalho uns dos outros e possam cobrir um ao outro caso alguém se ausente. Isso pode não ser mencionado durante uma entrevista, já que o grupo poderia não considerar uma parte integrante do seu trabalho.

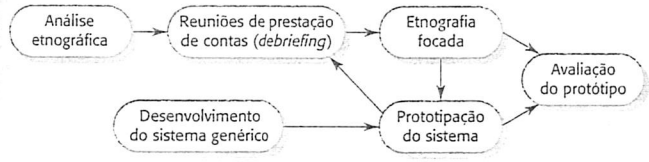
Suchman (1983) foi pioneira no uso da etnografia para estudar o trabalho de escritório. Ela constatou que as práticas de trabalho reais eram muito mais ricas, complexas e dinâmicas do que os modelos simples presumidos pelos sistemas de automação. A diferença entre o trabalho presumido e o real foi a razão mais importante para esses sistemas de escritório não produzirem um efeito significativo na produtividade. Crabtree (2003) discute uma ampla gama de estudos desde então e descreve, em geral, o uso da etnografia no projeto de sistemas. Em minha própria pesquisa, investiguei métodos de integração da etnografia nos processos de engenharia de software vinculando-a com os métodos de engenharia de requisitos (VILLER; SOMMERVILLE, 2000) e padrões de documentação da interação em sistemas cooperativos (MARTIN; SOMMERVILLE, 2004).

A etnografia é particularmente eficaz para descobrir dois tipos de requisitos:

1. Requisitos derivados da maneira que as pessoas realmente trabalham, e não da maneira que as definições de processos de negócio dizem que deveriam trabalhar. Na prática, as pessoas nunca seguem processos formais. Por exemplo, os controladores de tráfego aéreo podem desligar um sistema de alerta de conflitos que detecta aeronaves que estão em rotas de colisão, embora os procedimentos de controle normais especifiquem que tal sistema deva ser utilizado. O sistema de alerta de conflitos é sensível e emite alertas sonoros mesmo quando os aviões estão bem distantes. Os controladores consideram que isso os distrai e preferem outras alternativas para assegurar que os aviões não sigam trajetórias de voo conflitantes.
2. Requisitos derivados da cooperação e do conhecimento das atividades das outras pessoas. Por exemplo, os controladores de tráfego aéreo podem usar o conhecimento do trabalho dos demais controladores para prever o número de aeronaves que entrarão em seu setor. Depois, eles modificam suas estratégias de controle, dependendo da carga de trabalho prevista. Portanto, um sistema de controle de tráfego aéreo automatizado deve permitir que os controladores em um setor tenham alguma visibilidade do trabalho realizado nos setores adjacentes.

A etnografia pode ser combinada com o desenvolvimento de um protótipo do sistema (Figura 4.8) e informa o desenvolvimento do protótipo, de modo que sejam necessários menos ciclos de refinamento. Além disso, a prototipação permite dar foco à etnografia por identificar problemas e questões que depois poderão ser discutidas com o etnógrafo, que buscará respostas para essas perguntas durante a próxima fase de estudo do sistema (SOMMERVILLE *et al.*, 1993).

FIGURA 4.8 Etnografia e prototipação para análise de requisitos.



A etnografia é útil para compreender os sistemas existentes, mas essa compreensão nem sempre ajuda na inovação. Esta é particularmente relevante para o desenvolvimento de novos produtos. Alguns analistas sugeriram que a Nokia usava etnografia para descobrir como as pessoas utilizavam seus telefones e, com base nisso, desenvolver novos modelos de telefone; a Apple, por outro lado, ignorou o uso atual e revolucionou a indústria de celulares com a introdução do iPhone.

Os estudos etnográficos podem revelar detalhes críticos dos processos que muitas vezes passam despercebidos por outras técnicas de elicitação de requisitos. Entretanto, em virtude do foco no usuário final, essa abordagem não é eficaz para descobrir requisitos de empresas ou de áreas mais amplas, nem para sugerir inovações. Portanto, a etnografia é uma entre as várias técnicas de elicitação de requisitos.

4.3.2 Histórias e cenários

As pessoas acham mais fácil se identificar com exemplos da vida real do que com descrições abstratas. Elas não são boas para falar de requisitos de sistema. No entanto, podem ser capazes de descrever como lidam com determinadas situações

ou imaginar coisas que poderiam fazer com uma nova forma de trabalhar. As histórias e cenários são maneiras de capturar esse tipo de informação, que pode ser usada posteriormente ao entrevistar grupos de *stakeholders* para discutir o sistema com outros grupos e desenvolver requisitos de sistema mais específicos.

Histórias e cenários são essencialmente a mesma coisa. Trata-se de uma descrição de como o sistema pode ser utilizado em alguma tarefa em particular. Histórias e cenários descrevem o que as pessoas fazem, quais informações usam e produzem e quais sistemas podem adotar nesse processo. A diferença está no modo como as descrições são estruturadas e no nível de detalhe apresentado. As histórias são escritas como texto narrativo e apresentam uma descrição de alto nível do uso do sistema; os cenários normalmente são estruturados com informações específicas coletadas, como entradas e saídas. Considero as histórias eficazes para estabelecer o 'panorama geral'. Partes delas podem ser desenvolvidas em mais detalhes e representadas como cenários.

A Figura 4.9 é um exemplo de história que desenvolvi para entender os requisitos do ambiente de aprendizagem digital iLearn, que apresentei no Capítulo 1. Ela descreve uma situação em uma escola primária (ensino fundamental) em que o professor está usando o ambiente para apoiar os projetos dos alunos sobre a indústria pesqueira. Dá para ver que se trata de uma descrição de alto nível. Sua finalidade é facilitar a discussão sobre como o iLearn poderia ser utilizado e atuar como um ponto de partida para a elicitação dos requisitos do sistema.

FIGURA 4.9 A história de um usuário para o sistema iLearn.

Compartilhamento de imagens na sala de aula

Jack é um professor de escola primária em Ullapool (uma vila no norte da Escócia). Ele decidiu que um projeto de sala de aula deveria se concentrar na indústria pesqueira da região, examinando a história, o desenvolvimento e o impacto econômico da pesca. Como parte do projeto, ele pede que os alunos reúnam e compartilhem lembranças dos parentes, usem arquivos de jornal e coletem fotografias antigas relacionadas à pesca e às comunidades pesqueiras da região. Os alunos usam um *wiki* do iLearn para reunir histórias sobre pesca e o SCRAN (um site de recursos de história) para acessar os arquivos do jornal e as fotografias. No entanto, Jack também precisa de um site de compartilhamento de imagens, pois quer que os alunos troquem e comentem as fotos uns dos outros e coloquem no site as imagens escaneadas de fotografias antigas que possam ter em suas famílias.

Jack envia um e-mail para um grupo de professores de escola primária, do qual é membro, para ver se alguém pode recomendar um sistema adequado. Dois professores respondem e ambos sugerem que ele use o KidsTakePics, um site de compartilhamento de imagens que permite aos professores conferirem e moderarem o conteúdo. Como o KidsTakePics não é integrado ao serviço de autenticação do iLearn, ele cria uma conta de professor e uma conta de turma. Ele utiliza o serviço de configuração do iLearn para adicionar o KidsTakePics aos serviços visualizados pelos alunos em sua turma para que, quando fizerem o *login*, possam usar imediatamente o sistema para enviar fotos de seus celulares, tablets e computadores da sala de aula.

A vantagem das histórias é que todo mundo pode se identificar facilmente com elas. Achamos que essa abordagem é especialmente útil para obter informações de uma comunidade mais ampla do que poderíamos entrevistar na realidade. Disponibilizamos as histórias em um *wiki* e convidamos professores e alunos do país inteiro para comentá-las.

Essas histórias de mais alto nível não entram em detalhes sobre um sistema, mas podem ser desenvolvidas em cenários mais específicos. Os cenários são descrições de exemplos de sessões de interação do usuário. Acredito que seja melhor apresentar

os cenários de uma maneira estruturada, em vez de um texto narrativo. As histórias de usuário utilizadas nos métodos ágeis, como na Programação Extrema, são cenários narrativos e não histórias genéricas para ajudar a elicitar requisitos.

Um cenário começa com uma descrição da interação. Durante o processo de elicitação, são acrescentados detalhes para criar uma descrição completa dessa interação. De modo geral, um cenário pode incluir:

1. uma descrição do que o sistema e os usuários esperam quando o cenário se inicia;
2. uma descrição do fluxo normal dos eventos no cenário;
3. uma descrição do que pode dar errado e de como esses problemas podem ser enfrentados;
4. informações sobre outras atividades que poderiam ocorrer ao mesmo tempo;
5. uma descrição do estado do sistema quando o cenário termina.

Como exemplo de um cenário, a Figura 4.10 descreve o que acontece quando um aluno envia fotos para o sistema KidsTakePics, conforme explicado na Figura 4.9. A diferença fundamental entre esse e outros sistemas é que o professor modera as fotos enviadas para conferir se são adequadas ao compartilhamento.

Nota-se que essa é uma descrição muito mais detalhada do que a da história relatada na Figura 4.9 e, portanto, pode ser utilizada para propor requisitos do sistema iLearn. Assim como as histórias, os cenários podem ser empregados para facilitar discussões com os *stakeholders*, que às vezes podem ter maneiras diferentes de atingir o mesmo resultado.

FIGURA 4.10 Cenário para o envio de fotos para o KidsTakePics.

Enviar fotos para o KidsTakePics

Pressuposto inicial: Um usuário ou grupo de usuários tem uma ou mais fotografias digitais para serem enviadas para o site de compartilhamento de imagens. Essas fotos estão salvas em um tablet ou notebook. Eles fizeram o *login* no site KidsTakePics.

Normal: O usuário opta por enviar as fotos e é solicitado a ele que selecione as fotos no computador a serem enviadas e escolha o nome do projeto sob o qual as fotos serão armazenadas. Os usuários também devem ter a opção de digitar palavras-chave que deverão ser associadas a cada foto enviada. Essas fotos recebem um nome criado pela conjunção do nome do usuário com o nome do arquivo da foto no computador local.

No final do envio, o sistema manda automaticamente um e-mail para o moderador do projeto, pedindo-lhe que verifique o novo conteúdo, e gera uma mensagem na tela para o usuário dizendo que essa verificação foi feita.

O que pode dar errado: Nenhum moderador está associado ao projeto selecionado. Um e-mail é gerado automaticamente para o administrador da escola pedindo para nomear um moderador do projeto. Os usuários devem ser informados de um possível atraso no procedimento para tornar suas fotos visíveis.

Fotos com o mesmo nome já foram enviadas pelo mesmo usuário. O usuário deve ser questionado se deseja enviar novamente as fotos, renomeá-las ou cancelar seu envio. Se os usuários escolherem reenviar, os originais serão sobrescritos. Se optarem por renomear, um novo nome será gerado automaticamente acrescentando um número ao nome de arquivo existente.

Outras atividades: O moderador pode estar logado no sistema e aprovar as fotos à medida que forem enviadas.

Estado do sistema ao terminar: O usuário está logado. As fotos escolhidas foram enviadas e receberam o status de 'aguardando moderação'. As fotos estarão visíveis para o moderador e para o usuário que as enviou.

4.4 ESPECIFICAÇÃO DE REQUISITOS

A especificação de requisitos é o processo de escrever os requisitos de usuário e de sistema em um documento de requisitos. Em condições ideais, esses requisitos devem ser claros, inequívocos, fáceis de entender, completos e consistentes. Na prática, isso é quase impossível de alcançar. Os *stakeholders* interpretam os requisitos de maneiras diferentes e muitas vezes há conflitos e incoerências inerentes a eles.

Os requisitos de usuário quase sempre são escritos em linguagem natural, complementada por diagramas e tabelas apropriados no documento de requisitos. Os requisitos de sistema também podem ser escritos em linguagem natural, mas outras notações baseadas em formulários, gráficos ou modelos matemáticos do sistema também podem ser utilizadas. A Figura 4.11 resume as possíveis notações para escrever requisitos de sistema.

FIGURA 4.11 Notações para escrever requisitos de sistema.

Notação	Descrição
Sentenças em linguagem natural	Os requisitos são escritos usando frases numeradas em linguagem natural. Cada frase deve expressar um requisito.
Linguagem natural estruturada	Os requisitos são escritos em linguagem natural em um formulário ou *template*. Cada campo fornece informações sobre um aspecto do requisito.
Notações gráficas	Modelos gráficos, suplementados por anotações em texto, são utilizados para definir os requisitos funcionais do sistema. São utilizados com frequência os diagramas de casos de uso e de sequência da UML.
Especificações matemáticas	Essas notações se baseiam em conceitos matemáticos como as máquinas de estados finitos ou conjuntos. Embora essas especificações inequívocas possam reduzir a ambiguidade em um documento de requisitos, a maioria dos clientes não compreende uma especificação formal. Eles não conseguem averiguar se ela representa o que desejam e relutam em aceitar essa especificação como um contrato do sistema (discutirei essa abordagem no Capítulo 10, que aborda a dependabilidade do sistema).

Os requisitos de usuário de um sistema devem descrever os requisitos funcionais e não funcionais de modo que sejam compreensíveis para os usuários do sistema que não têm conhecimento técnico detalhado. Em condições ideais, eles devem especificar apenas o comportamento externo do sistema. O documento de requisitos não deve incluir detalhes da arquitetura ou do projeto (*design*) do sistema. Consequentemente, ao escrever requisitos de usuário, não se deve usar jargões de software, notações estruturadas ou notações formais. Os requisitos de usuário devem ser escritos em linguagem natural, com tabelas simples, formulários e diagramas intuitivos.

Os requisitos de sistema são versões ampliadas dos requisitos de usuário, que os engenheiros de software usam como ponto de partida para o projeto do sistema, acrescentando detalhes e explicando como o sistema deverá atender os requisitos de usuário. Eles podem ser utilizados como parte do contrato para a implementação do sistema, portanto devem ser uma especificação completa e detalhada do sistema inteiro.

Em condições ideais, os requisitos de sistema devem descrever apenas o comportamento externo do sistema e suas restrições operacionais. Eles não devem se preocupar com o modo que o sistema deve ser projetado ou implementado. No entanto, no nível de detalhe exigido para especificar completamente um sistema de software complexo, não é possível nem desejável excluir todas s informações de projeto (*design*). Existem várias razões para isso:

1. Pode ser necessário fazer o projeto de uma arquitetura inicial do sistema para ajudar a estruturar a especificação dos requisitos. Os requisitos de sistema são organizados de acordo com diferentes subsistemas que o compõem. Fizemos isso quando definimos os requisitos do sistema iLearn, no qual propusemos a arquitetura exibida na Figura 1.8.
2. Na maioria dos casos, os sistemas devem interoperar com os sistemas existentes, o que restringe o projeto e impõe requisitos ao novo sistema.
3. Pode ser necessário o uso de uma arquitetura específica para satisfazer requisitos não funcionais, como a programação N-versões — discutida no Capítulo 11 — para alcançar confiabilidade. Um regulador externo que precise certificar-se de que o sistema é seguro (*safe*) pode especificar que deve ser utilizado um projeto de arquitetura já certificado.

4.4.1 Especificação em linguagem natural

A linguagem natural tem sido utilizada para escrever requisitos de software desde os anos 1950. É uma linguagem expressiva, intuitiva e universal. Também é potencialmente vaga e ambígua, sendo que a sua interpretação depende da experiência do leitor. Consequentemente, tem havido muitas propostas de maneiras alternativas para escrever os requisitos. No entanto, nenhuma dessas propostas foi adotada amplamente, e a linguagem natural continuará sendo a maneira mais utilizada de especificar requisitos de sistema e software.

Para minimizar os mal-entendidos ao escrever requisitos em linguagem natural, recomendo seguir estas diretrizes simples:

1. Inventar um formato padrão e garantir que todas as definições de requisitos o sigam. Padronizar o formato diminui a probabilidade de omissões e torna os requisitos mais fáceis de serem conferidos. Sempre que for possível, sugiro escrever o requisito em uma ou duas frases de linguagem natural.
2. Usar a linguagem coerentemente para distinguir entre requisitos obrigatórios e desejáveis. Os requisitos obrigatórios são aqueles que o sistema deve apoiar – e normalmente são escritos usando 'deve'. Os requisitos desejáveis não são essenciais – e são escritos usando 'pode'.
3. Usar realce de texto (negrito, itálico ou cor) para destacar partes importantes do requisito.
4. Não supor que os leitores compreendem a linguagem técnica da engenharia de software. É fácil que palavras como 'arquitetura' e 'módulo' sejam mal compreendidas. Sempre que possível, evitar o uso de jargões, abreviações e acrônimos.
5. Sempre que possível, tentar associar um racional a cada requisito de usuário. O racional deve explicar por que o requisito foi incluído e quem o propôs (a origem do requisito), de modo que se saiba a quem recorrer se o requisito

precisar ser alterado. O racional dos requisitos é particularmente útil quando isso acontece, já que essa mudança pode ajudar a decidir quais alterações seriam indesejáveis.

A Figura 4.12 ilustra como essas diretrizes podem ser utilizadas. Ela inclui dois requisitos para software embarcado na bomba de insulina automatizada, introduzida no Capítulo 1. Outros requisitos desse sistema embarcado são definidos no documento de requisitos da bomba de insulina, que pode ser baixado no site do livro (em inglês).

FIGURA 4.12 Exemplos de requisitos do sistema de software da bomba de insulina.

3.2 O sistema deve medir o nível de açúcar no sangue e fornecer insulina, se for necessário, a cada 10 minutos. (*As variações do açúcar no sangue são relativamente lentas, então é desnecessário medir com uma frequência maior; a medição menos frequente poderia levar a níveis de açúcar sanguíneo desnecessariamente elevados.*)

3.6 O sistema deve executar uma rotina de autoteste a cada minuto com as condições a serem testadas e as ações associadas, definidas na Tabela 1 do documento de requisitos. (*Uma rotina de autoteste pode descobrir problemas de hardware e software e alertar o usuário de que a operação normal pode ser impossível.*)

4.4.2 Especificações estruturadas

A linguagem natural estruturada é uma maneira de escrever os requisitos de sistema, de modo que estes sejam escritos em uma forma padrão em vez de em texto livre. Essa abordagem mantém a maior parte da expressividade e da clareza da linguagem natural, mas garante que alguma uniformidade seja imposta à especificação. As notações que adotam linguagem estruturada usam modelos para especificar requisitos de sistema. Essa especificação pode usar construtos de linguagem de programação para mostrar alternativas e iteração, podendo destacar elementos-chave por intermédio de sombreamento ou de fontes diferentes.

Os Robertsons (ROBERTSON; ROBERTSON, 2013), em seu livro sobre o método VOLERE de engenharia de requisitos, recomendam que os requisitos de usuário sejam escritos inicialmente em cartões, com um requisito por cartão. Eles sugerem uma série de campos em cada cartão, como o racional dos requisitos, as dependências de outros requisitos, a origem dos requisitos e os materiais de apoio. Isso é similar à abordagem utilizada no exemplo de uma especificação estruturada, exibido na Figura 4.13.

Problemas com o uso da linguagem natural na especificação dos requisitos

A flexibilidade da linguagem natural, tão útil para a especificação, costuma causar problemas. Existe espaço para escrever requisitos obscuros e os leitores (os projetistas) podem interpretar erroneamente os requisitos porque eles e os usuários têm experiências diferentes. É fácil fundir vários requisitos em uma única frase, o que pode dificultar a estruturação dos requisitos em linguagem natural.

FIGURA 4.13 Especificação estruturada de um requisito para uma bomba de insulina.

Bomba de insulina/Software de controle/SRS/3.3.2	
Função	Computar a dose de insulina: nível de açúcar seguro.
Descrição	Computa a dose de insulina a ser fornecida quando o nível de açúcar atual estiver na zona segura entre 3 e 7 unidades.
Entradas	Leitura atual do açúcar (r2), as duas leituras prévias (r0 e r1).
Fonte	Leitura atual de açúcar do sensor. Outras leituras da memória.
Saídas	DoseComp – a dose de insulina a ser fornecida.
Destino	Laço de controle principal.
Ação	DoseComp é igual a zero se o nível de açúcar estiver estável ou caindo; ou se o nível estiver aumentando, mas a taxa de crescimento estiver diminuindo. Se o nível estiver aumentando e a taxa de crescimento também, então a DoseComp é obtida pela divisão por 4 da diferença entre o nível de açúcar atual e o nível anterior, arredondando o resultado. Se o resultado for arredondado para zero, então a DoseComp é definida como dose mínima que pode ser fornecida (ver Figura 4.14).
Requer	Duas leituras prévias para que a taxa de variação do nível de açúcar possa ser calculada.
Pré-condição	O reservatório de insulina contém pelo menos a dose máxima permitida.
Pós-condição	r0 é substituída por r1, então r1 é substituída por r2.
Efeitos colaterais	Nenhum.

Para usar uma abordagem estruturada para especificar requisitos de sistema, é preciso definir um ou mais *templates* para os requisitos e representá-los como formulários estruturados. A especificação pode ser estruturada em volta dos objetos manipulados pelo sistema, das funções realizadas por ele ou dos eventos processados. Um exemplo de especificação baseada em formulário, nesse caso, é o que define como calcular a dose de insulina a ser fornecida quando o açúcar no sangue estiver dentro da faixa segura, como mostra a Figura 4.13.

Quando um *template* é empregado para especificar requisitos funcionais, as seguintes informações devem ser incluídas:

1. uma descrição da função ou entidade que está sendo especificada;
2. uma descrição das entradas e suas origens;
3. uma descrição das saídas e sua destinação;
4. informações sobre os dados necessários para computar ou outras entidades no sistema que sejam necessárias (a parte 'requer');
5. uma descrição da ação a ser tomada;
6. se for utilizada uma abordagem funcional, uma precondição estabelecendo o que deve ser verdadeiro antes da função ser invocada e uma pós-condição especificando o que é verdadeiro após a função ser invocada;
7. uma descrição dos efeitos colaterais (se houver) da operação.

O uso de especificações estruturadas remove alguns dos problemas da especificação em linguagem natural. A variabilidade na especificação é reduzida, e os requisitos são organizados com mais eficácia. No entanto, às vezes é difícil escrever os requisitos de uma maneira clara e inequívoca, particularmente quando computações complexas (como calcular a dose de insulina) devem ser especificadas.

Para resolver esse problema, é possível acrescentar mais informações aos requisitos em linguagem natural, por exemplo, usando tabelas ou modelos gráficos do sistema. Esses recursos podem mostrar como os cálculos são feitos, como o estado do sistema muda, como os usuários interagem com o sistema e como as sequências de ações são realizadas.

As tabelas são particularmente úteis quando existe uma série de possíveis situações alternativas e é preciso descrever as ações a serem tomadas para cada uma delas. A bomba de insulina baseia seus cálculos do requisito de insulina na taxa de variação dos níveis de açúcar no sangue. Essas taxas são calculadas usando as leituras atual e prévia. A Figura 4.14 é uma descrição tabular de como a taxa de variação do açúcar no sangue é utilizada para calcular a quantidade de insulina a ser fornecida.

FIGURA 4.14 Especificação tabular do cálculo em uma bomba de insulina.

Condição	Ação
Nível de açúcar em queda (r2 < r1)	DoseComp = 0
Nível de açúcar estável (r2 = r1)	DoseComp = 0
Nível de açúcar em alta e taxa de crescimento em queda ((r2 – r1) < (r1 – r0))	DoseComp = 0
Nível de açúcar em alta e taxa de crescimento estável ou em alta r2 > r1 & ((r2 – r1) ≥ (r1 – r0))	DoseComp = arredondar ((r2 – r1) / 4) Se resultado arredondado = 0, então DoseComp = DoseMínima

4.4.3 Casos de uso

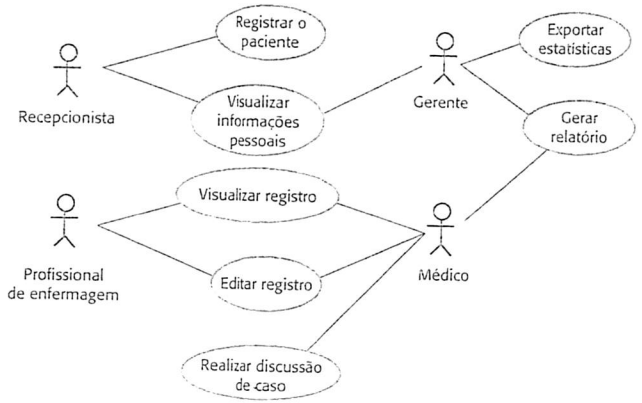
Os casos de uso são uma maneira de descrever as interações entre usuários e um sistema usando um modelo gráfico e um texto estruturado. Foram introduzidos pela primeira vez no método Objectory (JACOBSON *et al.*, 1993) e hoje se tornaram uma característica fundamental da UML. Em sua forma mais simples, um caso de uso identifica os atores envolvidos em uma interação e nomeia o tipo de interação. Depois, são adicionadas informações descrevendo a interação com o sistema, que pode ser uma descrição textual ou um ou mais modelos gráficos — como os diagramas de sequência ou de máquina de estados da UML (ver Capítulo 5).

Os casos de uso são documentados por meio de um diagrama de casos de uso de alto nível. O conjunto de casos de uso representa todas as interações possíveis que serão descritas nos requisitos de sistema. Os atores no processo, que podem ser seres humanos ou outros sistemas, são representados como 'bonecos palito'. Cada classe de interação é representada como uma elipse nomeada. Linhas fazem a ligação entre os atores e a interação. Opcionalmente, pontas de seta podem ser acrescentadas às linhas para mostrar como a interação começa. Isso é ilustrado pela Figura 4.15, que mostra alguns dos casos de uso do sistema Mentcare.

Os casos de uso identificam cada interação entre o sistema e seus usuários ou outros sistemas. Cada caso de uso deve ser documentado com uma descrição textual, que pode ser ligada a outros modelos — também em UML — para compor um cenário mais detalhado. Por exemplo, uma descrição resumida do uso de caso de Realizar discussão de caso da Figura 4.15 poderia ser:

Realizar discussão de caso permite que dois ou mais médicos, trabalhando em consultórios diferentes, vejam o registro do mesmo paciente ao mesmo tempo. Um médico inicia a discussão do caso de um paciente escolhendo as pessoas envolvidas em um menu suspenso de médicos que estão on-line. *O registro do paciente é exibido em suas telas, mas apenas o médico que iniciou a consulta pode editar o registro. Além disso, cria-se um chat para ajudar a coordenar as ações. Presume-se que uma chamada telefônica ou comunicação por voz possa ser providenciada separadamente.*

FIGURA 4.15 Casos de uso do sistema Mentcare.



A UML é um padrão para modelagem orientada a objetos, então os casos de uso e a elicitação de requisitos baseada em casos de uso são utilizadas no processo de engenharia de requisitos. No entanto, minha experiência com os casos de uso é que eles são muito refinados para serem úteis na discussão de requisitos. Os *stakeholders* não compreendem o termo *caso de uso*, não acham útil o modelo gráfico e, muitas vezes, não estão interessados em uma descrição detalhada de cada interação do sistema. Consequentemente, acho os casos de uso mais úteis no projeto de sistemas do que na engenharia de requisitos. Discutirei melhor esse assunto no Capítulo 5, que mostra como os casos de uso são utilizados com outros modelos de sistema para documentar um projeto (*design*).

Algumas pessoas acham que cada caso de uso é um cenário de interação único e detalhado. Outras, como Stevens e Pooley (2006), sugerem que cada caso inclui um conjunto relacionado de cenários detalhados. Cada um deles é um único caminho do caso de uso. Portanto, haveria um cenário para a interação normal, além de cenários para cada exceção possível. Na prática, dá para usá-los de ambas as formas.

4.4.4 O documento de requisitos de software

O documento de requisitos de software (às vezes chamado de especificação de requisitos de software ou ERS) é uma declaração oficial do que os desenvolvedores do sistema devem implementar. Ele pode incluir os requisitos de usuário para um sistema e uma especificação detalhada dos requisitos de sistema. Às vezes, os requisitos de usuário e de sistema são integrados em uma descrição única. Em outros casos, os requisitos de usuário são descritos em um capítulo introdutório na especificação de requisitos de sistema.

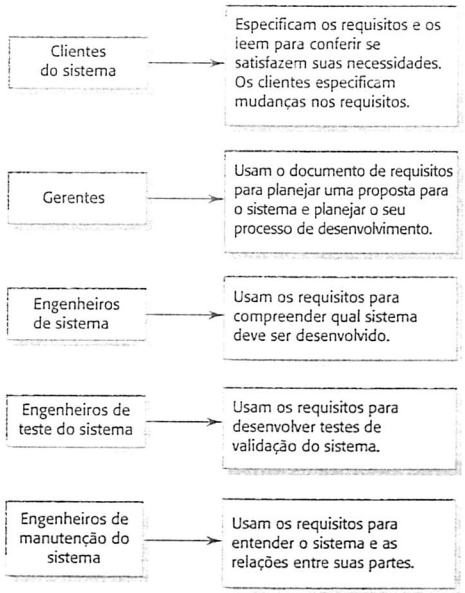
Os documentos de requisito são essenciais quando: os sistemas têm o seu desenvolvimento terceirizado, times diferentes desenvolvem partes diferentes do sistema ou uma análise detalhada dos requisitos é obrigatória. Em outras circunstâncias, como o desenvolvimento de um produto de software ou de um sistema de negócio, um documento de requisitos detalhado pode não ser necessário.

Os métodos ágeis argumentam que os requisitos mudam com tanta rapidez que um documento de requisitos fica obsoleto logo que é escrito, então o esforço é quase todo desperdiçado. Em vez de um documento formal, as abordagens ágeis costumam coletar os requisitos de usuário de modo incremental e escrevê-los em cartões ou lousas na forma de pequenas histórias de usuário. Então, o usuário priorizará essas histórias para implementação nos incrementos seguintes do sistema.

Nos sistemas de negócio nos quais os requisitos são instáveis, creio que essa abordagem é boa. No entanto, ainda acredito que seja útil escrever um documento de suporte resumido que defina o negócio e os requisitos de dependabilidade do sistema; é fácil esquecer os requisitos que se aplicam ao sistema como um todo quando nos concentramos nos requisitos funcionais da próxima versão do sistema.

O documento de requisitos tem um conjunto de usuários diversos, variando da alta gerência da organização que está pagando pelo sistema até os engenheiros responsáveis por desenvolver o software. A Figura 4.16 mostra os possíveis usuários do documento e como eles o utilizam.

FIGURA 4.16 Usuários de um documento de requisitos.



A diversidade dos possíveis usuários significa que o documento de requisitos tem de ser acordado. Ele deve descrever os requisitos para os clientes, defini-los em detalhes precisos para desenvolvedores e testadores, bem como incluir informações sobre futuras evoluções do sistema. As informações sobre mudanças antecipadas ajudam os projetistas do sistema a evitar decisões de projeto restritivas e os engenheiros de manutenção a adaptar o sistema aos novos requisitos.

O nível de detalhe que deve ser incluído em um documento de requisitos depende do tipo de sistema que está sendo desenvolvido e do processo de desenvolvimento

utilizado. Os sistemas críticos precisam de requisitos detalhados porque a segurança (*safety*) e a segurança da informação (*security*) devem de ser analisadas em detalhes, a fim de encontrar possíveis erros nos requisitos. Quando o sistema é desenvolvido por uma empresa diferente (por meio de terceirização, por exemplo), as especificações do sistema precisam ser detalhadas e precisas. Se o desenvolvimento for interno, usando um processo de desenvolvimento iterativo, o documento de requisitos pode ser menos detalhado. Podem ser acrescentados detalhes aos requisitos e as ambiguidades resolvidas durante o desenvolvimento do sistema.

A Figura 4.17 mostra uma possível organização do documento de requisitos, baseada em um padrão do IEEE para esse tipo de documento (IEEE, 1998). Esse padrão é genérico e pode ser adaptado a usos específicos. Nesse caso, o padrão precisa ser ampliado para incluir informações sobre a evolução prevista para o sistema, pois elas ajudam os responsáveis pela manutenção do sistema e permitem que os projetistas incluam suporte para futuras características do sistema.

FIGURA 4.17 Estrutura de um documento de requisitos.

Capítulo	Descrição
Prefácio	Define o público-alvo do documento e descreve seu histórico de versões, incluindo a fundamentação para a criação de uma nova versão e um resumo das mudanças feitas em cada uma.
Introdução	Descreve a necessidade do sistema. Deve descrever resumidamente as funções do sistema e explicar como ele vai trabalhar com outros sistemas. Também precisa descrever como o sistema se encaixa nos objetivos de negócio gerais ou estratégicos da organização que contratou o software.
Glossário	Define os termos técnicos utilizados no documento. Deve-se evitar fazer pressupostos sobre a experiência ou a especialização do leitor.
Definição dos requisitos de usuário	Descreve os serviços fornecidos para o usuário. Os requisitos não funcionais do sistema também devem ser descritos nesta seção. Essa descrição pode usar linguagem natural, diagramas ou outras notações compreensíveis para os clientes. Os padrões de produto e processo que devem ser seguidos têm de ser especificados.
Arquitetura do sistema	Esse capítulo apresenta uma visão geral e de alto nível da arquitetura prevista para o sistema, mostrando a distribuição das funções pelos módulos do sistema. Os componentes de arquitetura reusados devem ser destacados.
Especificação dos requisitos de sistema	Descreve os requisitos funcionais e não funcionais em mais detalhes. Se for necessário, mais detalhes também são acrescentados aos requisitos não funcionais. Podem ser definidas interfaces com outros sistemas.
Modelos do sistema	Esse capítulo inclui modelos gráficos do sistema, mostrando as relações entre os componentes do sistema e entre o sistema e seu ambiente. Exemplos possíveis são os modelos de objeto, modelos de fluxo de dados ou modelos semânticos de dados.
Evolução do sistema	Descreve os pressupostos fundamentais nos quais o sistema se baseia e quaisquer mudanças previstas em virtude da evolução do hardware, da mudança nas necessidades dos usuários etc. Essa seção é útil para os projetistas do sistema, já que pode ajudá-los a evitar decisões de projeto que restringiriam futuras mudanças prováveis no sistema.
Apêndices	Fornecem informações específicas, detalhadas, relacionadas à aplicação que está sendo desenvolvida — por exemplo, descrições de hardware e banco de dados. Os requisitos de hardware definem as configurações mínima e ideal do sistema; os requisitos de banco de dados definem a organização lógica dos dados utilizados pelo sistema e seus relacionamentos.
Índice	Vários índices para o documento podem ser incluídos, bem como índice alfabético normal, índice de diagramas, índice de funções etc.

Naturalmente, as informações incluídas em um documento de requisitos dependem do tipo de software que está sendo desenvolvido e da abordagem que está sendo utilizada para o desenvolvimento. Um documento de requisitos com uma estrutura parecida com a da Figura 4.17 poderia ser produzido para um sistema de engenharia complexo que inclua hardware e software desenvolvidos por empresas diferentes. O

documento de requisitos tende a ser longo e detalhado. Portanto, é importante incluir uma tabela de conteúdo abrangente e um índice do documento para que os leitores possam encontrar facilmente as informações necessárias.

Por outro lado, o documento de requisitos de um produto de software desenvolvido internamente vai deixar de fora muitos dos capítulos detalhados sugeridos anteriormente. O foco estará na definição dos requisitos de usuário e dos requisitos de sistema não funcionais de alto nível. Os projetistas e os programadores do sistema devem usar de seu bom senso para decidir como atender a descrição dos requisitos de usuário do sistema.

Padrões de documento de requisitos

Um grande número de organizações, como o Departamento de Defesa dos Estados Unidos e o IEEE, definiu padrões para documentos de requisito. Esses padrões normalmente são genéricos, contudo são úteis como uma base para desenvolver padrões organizacionais mais detalhados. O IEEE é um dos fornecedores de padrão mais conhecidos e desenvolveu um para a estrutura dos documentos de requisitos. Esse padrão é mais adequado para sistemas como comando e controle militar, que têm uma vida útil longa e geralmente são desenvolvidos por um grupo de organizações.

4.5 VALIDAÇÃO DE REQUISITOS

A validação de requisitos é o processo de conferir se os requisitos definem o sistema que o cliente realmente quer. Ele se sobrepõe à elicitação e à análise, já que é voltado para encontrar problemas. A validação de requisitos é criticamente importante porque os erros em um documento de requisitos podem levar a grandes custos de retrabalho quando esses problemas são descobertos durante o desenvolvimento ou após o sistema entrar em serviço.

O custo de corrigir um problema nos requisitos com uma alteração no sistema normalmente é muito maior do que o de consertar erros de projeto ou de código. Uma mudança nos requisitos significa, geralmente, que o projeto e a implementação do sistema também deverão ser modificados. Além disso, o sistema deve ser reiniciado.

Durante o processo de validação de requisitos, diferentes tipos de conferências devem ser executados nos requisitos do documento. Essas conferências incluem:

1. *Conferência da validade.* Confere se os requisitos refletem as reais necessidades dos usuários do sistema. Em virtude da mudança das circunstâncias, os requisitos de usuário podem ter mudado desde o que foi originalmente elicitado.
2. *Conferência da consistência.* Os requisitos no documento não devem entrar em conflito entre si, isto é, não deve haver restrições contraditórias ou descrições diferentes da mesma função do sistema.
3. *Conferência da completude.* O documento de requisitos deve incluir aqueles que definem todas as funções e as restrições pretendidas pelo usuário do sistema.
4. *Conferência do realismo.* Usando o conhecimento das tecnologias existentes, os requisitos devem ser conferidos para assegurar que possam ser implementados dentro do orçamento proposto para o sistema. Essas conferências também devem levar em conta o orçamento e o cronograma de desenvolvimento do sistema.
5. *Verificabilidade.* Para diminuir o potencial de conflito entre o cliente e o contratante, os requisitos do sistema sempre devem ser escritos de modo que sejam verificáveis. Isso significa ser capaz de escrever um conjunto de testes que possam demonstrar que o sistema entregue satisfaz cada um dos requisitos especificados.

Revisões de requisitos

Uma revisão de requisitos é um processo no qual um grupo de pessoas relacionadas ao cliente do sistema e o desenvolvedor do sistema leem o documento de requisitos em detalhes e averiguam erros, anomalias e inconsistências. Depois de detectados e registrados, cabe ao cliente e ao desenvolvedor negociarem de que modo os problemas identificados devem ser solucionados.

Uma série de técnicas de validação de requisitos pode ser utilizada individualmente ou em conjunto:

1. *Revisões de requisitos.* Os requisitos são analisados sistematicamente por uma equipe de revisores que conferem erros e inconsistências.
2. *Prototipação.* Isso envolve o desenvolvimento de um modelo executável do sistema e o uso desse modelo com os usuários finais e clientes para ver se satisfaz suas necessidades e expectativas. Os *stakeholders* experimentam o sistema e opinam sobre mudanças nos requisitos para o time de desenvolvimento.
3. *Geração de casos de teste.* Os requisitos devem ser testáveis. Se os testes dos requisitos forem concebidos como parte do processo de validação, frequentemente isso revela problemas nos requisitos. Se um teste for difícil ou impossível de projetar, normalmente isso significa que os requisitos serão difíceis de implementar e devem ser reconsiderados. Desenvolver testes a partir dos requisitos de usuário antes de qualquer código ser escrito faz parte do desenvolvimento dirigido por testes.

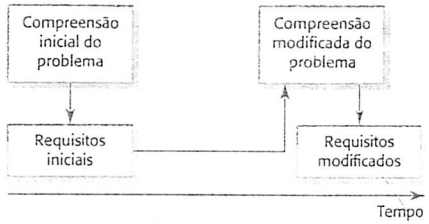
Não se deve subestimar os problemas envolvidos na validação dos requisitos. No final das contas, é difícil mostrar que um conjunto de requisitos satisfaz de fato as necessidades de um usuário. Os usuários precisam imaginar o sistema em operação e como ele se encaixaria em seu trabalho; até para os profissionais de informática qualificados é difícil realizar esse tipo de análise abstrata, e é ainda mais para os usuários do sistema.

Como consequência, é raro encontrar todos os problemas de requisitos durante o processo de validação. Inevitavelmente, serão necessárias outras alterações nos requisitos para corrigir omissões e mal-entendidos, mesmo após chegar a um consenso no documento de requisitos.

4.6 MUDANÇA DE REQUISITOS

Os requisitos dos sistemas de software grandes sempre estão mudando. Uma razão para as mudanças frequentes é que esses sistemas são desenvolvidos para tratar de problemas 'traiçoeiros' — problemas que não podem ser definidos completamente (RITTEL; WEBBER, 1973). Como o problema não pode ser totalmente definido, os requisitos de software obrigatoriamente são incompletos. Durante o processo de desenvolvimento de software, a compreensão que os *stakeholders* têm do problema muda constantemente (Figura 4.18). Os requisitos do sistema devem evoluir para refletir essa compreensão alterada do problema.

FIGURA 4.18 Evolução dos requisitos.



Depois que o sistema foi instalado e está sendo utilizado normalmente, é inevitável que surjam novos requisitos. Em parte, essa é uma consequência de erros e omissões nos requisitos originais, que precisam ser corrigidos. No entanto, a maioria das mudanças nos requisitos do sistema surge em razão de mudanças no ambiente de negócios do sistema:

1. Os ambientes de negócios e de tecnologia sempre mudam após a instalação do sistema. Pode ser introduzido um novo hardware e o atual pode ser atualizado. Pode ser necessário fazer a interface do sistema com outros sistemas. As prioridades do negócio podem mudar (com consequentes mudanças no suporte de sistema exigido), e novas legislações e normas podem ser introduzidas, exigindo adequação e conformidade do sistema.
2. As pessoas que pagam por um sistema e as pessoas que usam o sistema raramente são as mesmas. Os clientes do sistema impõem requisitos em virtude das restrições organizacionais e orçamentárias, que podem entrar em conflito com os requisitos dos usuários finais; após a entrega, novas características podem acabar tendo de ser adicionadas para dar suporte ao usuário, a fim de que o sistema cumpra suas metas.
3. Normalmente, os grandes sistemas têm uma comunidade variada de *stakeholders*, cada uma com diferentes requisitos e com prioridades que podem ser conflitantes ou contraditórias. Inevitavelmente, os requisitos do sistema final envolvem uma conciliação, e alguns *stakeholders* devem ter prioridade. Com a experiência, muitas vezes se descobre que a igualdade de suporte dado a diferentes *stakeholders* deve ser modificada e que os requisitos devem ser priorizados novamente.

Como os requisitos estão evoluindo, é preciso acompanhar cada um e manter vínculos entre os requisitos dependentes para que se possa avaliar o impacto das mudanças. Portanto, um processo formal é necessário para propor mudanças e vinculá-las aos requisitos do sistema. Esse processo de 'gerenciamento de requisitos' deve começar logo que uma versão de rascunho do documento de requisitos estiver disponível.

Os processos de desenvolvimento ágil foram concebidos para lidar com requisitos que mudam durante o processo de desenvolvimento. Nesses processos, quando um usuário propõe uma mudança nos requisitos, ela não passa por um processo formal de gerenciamento de mudanças. Em vez disso, o usuário tem de priorizar a mudança e, se for de alta prioridade, decidir quais características do sistema que foram planejadas para a próxima iteração devem ser abandonadas para que ela seja implementada.

Requisitos duradouros e voláteis

Alguns requisitos são mais susceptíveis à mudança do que outros. Os requisitos duradouros são aqueles associados às atividades centrais da organização, que mudam lentamente. Os requisitos duradouros estão associados com atividades de trabalho fundamentais. Os requisitos voláteis são mais propensos à mudança. Normalmente, estão associados às atividades de apoio que refletem como a organização faz o seu trabalho, em vez de associados ao trabalho em si.

O problema com essa abordagem é que os usuários não são necessariamente as pessoas mais indicadas para decidir se a mudança de um requisito tem ou não um custo-benefício justificável. Nos sistemas com vários *stakeholders*, as mudanças vão beneficiar alguns e outros não. Muitas vezes, é melhor para uma autoridade independente, que pode equilibrar as necessidades de todos os *stakeholders*, decidir sobre as mudanças que deveriam ser aceitas.

4.6.1 Planejamento do gerenciamento de requisitos

O planejamento do gerenciamento de requisitos tem a ver com estabelecer como será gerenciado um conjunto de requisitos em evolução. Durante o estágio de planejamento, é preciso decidir sobre uma série de questões:

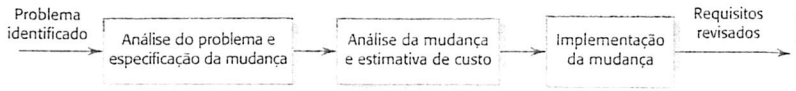
1. *Identificação dos requisitos.* Cada requisito deve ser identificado para que possa ser referido por outros requisitos e utilizado em avaliações de rastreabilidade.
2. *Processo de gerenciamento de mudança.* Esse é um conjunto de atividades que avaliam o impacto e o custo das mudanças. Discutirei esse processo em mais detalhes na seção a seguir.
3. *Políticas de rastreabilidade.* Definem os relacionamentos que devem ser registrados entre cada requisito e entre os requisitos e o projeto (*design*) do sistema. A política de rastreabilidade também deve definir como esses registros devem ser mantidos.
4. *Apoio de ferramentas.* O gerenciamento de requisitos envolve o processamento de grande quantidade de informações sobre os requisitos. As ferramentas que podem ser utilizadas variam de sistemas especializados no gerenciamento de requisitos até planilhas compartilhadas e sistemas de bancos de dados simples.

O gerenciamento de requisitos precisa de suporte automatizado, e as ferramentas de software para isso devem ser escolhidas durante a fase de planejamento. É preciso de apoio de ferramentas para:

1. *Armazenamento de requisitos.* Os requisitos devem ser mantidos em um repositório de dados gerenciado e seguro, que seja acessível a todos os envolvidos no processo de engenharia de requisitos.
2. *Gerenciamento de mudança.* O processo de gerenciamento de mudança (Figura 4.19) é simplificado se houver apoio ativo de ferramentas, que podem acompanhar as mudanças sugeridas e as respostas a essas sugestões.
3. *Gerenciamento da rastreabilidade.* Conforme discutido acima, o apoio de ferramentas para rastreabilidade permite a descoberta de requisitos relacionados. Existem algumas ferramentas que usam técnicas de processamento

de linguagem natural para ajudar a descobrir possíveis relacionamentos entre os requisitos.

FIGURA 4.19 Gerenciamento de mudança de requisitos.



Nos sistemas pequenos, não é preciso usar ferramentas especializadas em gerenciamento de requisitos. Ele pode ser realizado usando documentos web, planilhas e bancos de dados compartilhados. No entanto, nos sistemas maiores, o apoio de ferramentas mais especializadas — sistemas como o DOORS (IBM, 2013) — facilita bastante o acompanhamento de uma grande quantidade de mudanças nos requisitos.

Gerenciamento de mudança de requisitos

O gerenciamento de mudança de requisitos (Figura 4.19) deve ser aplicado a todas as mudanças propostas para os requisitos de um sistema após a aprovação do documento de requisitos. O gerenciamento de mudança é essencial para decidir se os benefícios de implementar novos requisitos são justificados pelos custos de sua implementação. A vantagem de usar um processo formal no gerenciamento de mudança é que todas as propostas de mudança são tratadas consistentemente e as mudanças no documento de requisitos são feitas de maneira controlada.

Existem três estágios principais em um processo de gerenciamento de mudança:

1. *Análise do problema e especificação da mudança.* O processo começa com a identificação de um problema de requisito ou, às vezes, com uma proposta de mudança específica. Durante esse estágio, o problema – ou a proposta de mudança – é analisado para averiguar se é válido. Essa análise é transmitida de volta para o requisitante da mudança, que pode responder com uma proposta de mudança de requisitos mais específica ou decidir pela desistência da solicitação.
2. *Análise da mudança e estimativa de custo.* O efeito da mudança proposta é avaliado com base nas informações de rastreabilidade e no conhecimento geral dos requisitos do sistema. O custo de fazer a mudança é estimado em termos de modificações nos documentos de requisitos e, se for apropriado, de projeto e de implementação do sistema. Depois de concluída a análise, toma-se uma decisão de proceder ou não com a mudança de requisitos.
3. *Implementação da mudança.* Os documentos de requisitos e, se for apropriado, de projeto e de implementação do sistema são modificados. É preciso organizar o documento de requisitos para que se possa fazer mudanças nele sem ter de reescrevê-lo ou reorganizá-lo amplamente. Assim como acontece com os programas, a facilidade de mudança dos documentos se obtém ao minimizar as referências externas e ao tornar as seções do documento as mais modulares possíveis. Desse modo, cada seção pode ser modificada e substituída sem afetar outras partes do documento.

Rastreabilidade de requisitos

Você precisa acompanhar as relações entre os requisitos, suas fontes e o projeto do sistema para que possa analisar as razões das alterações propostas e o impacto que essas mudanças tendem a ter em outras partes do sistema. Você precisa também ser capaz de rastrear como uma mudança atravessa o sistema. Por quê?

Se um novo requisito tiver de ser implementado com urgência, sempre existe a tentação de mudar o sistema e depois modificar retrospectivamente o documento de requisitos. Quase inevitavelmente isso coloca em descompasso a especificação dos requisitos e a implementação do sistema. Depois de feitas as mudanças, é fácil esquecer de incluí-las no documento de requisitos. Em algumas circunstâncias, é preciso realizar mudanças emergenciais em um sistema. Nesses casos, é importante atualizar o documento de requisitos o quanto antes para incluir os requisitos revisados.

PONTOS-CHAVE

- Os requisitos de um sistema de software estabelecem o que o sistema deve fazer e definem as restrições à sua operação e implementação.
- Os requisitos funcionais são declarações de serviços que o sistema deve prestar ou descrições de como devem ser feitas algumas computações.
- Os requisitos não funcionais costumam restringir o sistema que está sendo desenvolvido e seu processo de desenvolvimento, sejam requisitos de produto, de negócios ou externos. Muitas vezes, estão relacionados com as propriedades emergentes do sistema e, portanto, se aplicam ao sistema como um todo.
- O processo de engenharia de requisitos inclui a elicitação, a especificação, a validação e o gerenciamento de requisitos.
- A elicitação de requisitos é um processo iterativo que pode ser representado como uma espiral de atividades — descoberta, classificação e organização, negociação e documentação dos requisitos.
- A especificação de requisitos é um processo de documentar formalmente os requisitos de usuário e de sistema e de criar um documento de requisitos de software.
- O documento de requisitos de software é uma declaração de acordo dos requisitos de sistema. Deve ser organizado para que os clientes do sistema e os desenvolvedores de software possam usá-lo.
- A validação de requisitos é o processo de conferir sua validade, consistência, completude, realismo e verificabilidade.
- As mudanças no ambiente de negócios, nas organizações e nas tecnologias levam, inevitavelmente, a mudanças nos requisitos de um sistema de software. O gerenciamento de requisitos é o processo de gerenciar e controlar essas mudanças.

LEITURAS ADICIONAIS

"Integrated requirements engineering: a tutorial." Esse é um tutorial que discute as atividades da engenharia de requisitos e como elas podem ser adaptadas para se encaixar na prática moderna de engenharia de software. SOMMERVILLE, I. *IEEE Software*, v. 22, n. 1, jan./fev. 2005. doi:10.1109/MS.2005.13>.

"Research directions in requirements engineering." É um bom levantamento da pesquisa em engenharia de software que destaca os futuros desafios da pesquisa na área para tratar dos problemas de escala e agilidade. CHENG, B. H. C.; ATLEE, J. M. *Proc. Conf. on Future of Software Engineering*, IEEE Computer Society, 2007. doi:10.1109/FOSE.2007.17.

Mastering the requirements process. 3. ed. Um livro bem escrito e fácil de ler baseado em um método particular (VOLERE), mas que também inclui muitos bons conselhos gerais sobre engenharia de requisitos. ROBERTSON, S.; ROBERTSON, J. Addison-Wesley, 2013.

SITE[3]

Apresentações em PowerPoint para este capítulo disponíveis em: <http://software-engineering-book.com/slides/chap4/>.

Links para vídeos de apoio disponíveis em: <http://software-engineering-book.com/videos/requirements-and-design/>.

Documento de requisitos da bomba de insulina disponível em: <https://iansommerville.com/software-engineering-book/case-studies/a-personal-insulin-pump/>.

Informação dos requisitos do sistema Mentcare disponível em: <https://iansommerville.com/software-engineering-book/case-studies/the-mentcare-system/>.

3 Todo o conteúdo disponibilizado na seção *Site* de todos os capítulos está em inglês.

EXERCÍCIOS

4.1 Identifique e descreva resumidamente quatro tipos de requisitos que podem ser definidos para um sistema baseado em computador.

4.2 Descubra ambiguidades ou omissões na seguinte declaração de requisitos de parte de um sistema emissor de bilhetes:

> *Uma máquina de emitir bilhetes vende passagens de trem. Os usuários escolhem seu destino e fornecem um cartão de crédito e um número de identificação pessoal. A passagem de trem é emitida e o cartão de crédito dos usuários é cobrado. Quando os usuários pressionam o botão de iniciar, ativam um menu de possíveis destinos, junto com uma mensagem para a escolha de um destino e do tipo de bilhete necessário. Depois que o destino é selecionado, o preço do bilhete é exibido e os clientes são solicitados a fornecer seu cartão de crédito. Sua validade é conferida e os usuários devem fornecer seu identificador pessoal (PIN). Quando a transação de crédito é validada, o bilhete é emitido.*

4.3 Reescreva a descrição acima usando a abordagem estruturada descrita neste capítulo. Resolva as ambiguidades identificadas de maneira coerente.

4.4 Escreva um conjunto de requisitos não funcionais para o sistema emissor de bilhetes, estabelecendo a confiabilidade esperada e o tempo de resposta.

4.5 Usando a técnica sugerida aqui, em que as descrições em linguagem natural são apresentadas em um formato padrão, escreva requisitos de usuário plausíveis para as seguintes funções:

- Um sistema de bomba de gasolina sem frentista que inclui uma leitora de cartão de crédito. O cliente passa o cartão na leitora, depois especifica a quantidade de combustível necessária. O combustível é fornecido e a conta do cliente é debitada.
- A função de liberação de dinheiro em um caixa eletrônico de banco.
- Em um sistema de *internet banking*, um recurso que permite aos clientes transferirem fundos de uma conta no banco para outra conta no mesmo banco.

4.6 Sugira como um engenheiro responsável pela elaboração de uma especificação de requisitos de sistema poderia acompanhar as relações entre os requisitos funcionais e não funcionais.

4.7 Usando seu conhecimento de como se utiliza um caixa eletrônico, desenvolva um conjunto de casos de uso que poderia servir como base para entender os requisitos de um sistema desse tipo.

4.8 Quem deveria estar envolvido em uma revisão de requisitos? Desenhe um modelo de processo mostrando como uma revisão de requisitos poderia ser organizada.

4.9 Quando são feitas alterações emergenciais nos sistemas, o software do sistema pode necessitar de alterações antes das mudanças nos requisitos terem sido aprovadas. Sugira um modelo de processo para fazer essas modificações que venha a garantir que o documento de requisitos e a implementação do sistema não fiquem incoerentes.

4.10 Você assumiu um emprego com um usuário de software que contratou seu antigo patrão para desenvolver um sistema para ele. Você descobre que a interpretação dos requisitos feita pela sua empresa é diferente da interpretação feita pelo antigo patrão. Discuta o que você deve fazer nessa situação. Você sabe que os custos para o seu atual patrão vão aumentar se as ambiguidades não forem resolvidas. No entanto, você também tem responsabilidade com seu antigo patrão no que diz respeito à confidencialidade.

REFERÊNCIAS

CRABTREE, A. *Designing collaborative systems*: a practical guide to ethnography. London: Springer-Verlag, 2003.

DAVIS, A. M. *Software requirements*: objects, functions and states. Englewood Cliffs, NJ: Prentice-Hall, 1993.

IBM. "Rational Doors next generation: requirements engineering for complex systems." *IBM Jazz Community Site*, 2013. Disponível em: <https://jazz.net/products/rational-doors-next-generation/>. Acesso em: 5 abr. 2018.

IEEE. "IEEE Recommended Practice for Software Requirements Specifications." In: _____. *IEEE software engineering standards collection*. Los Alamitos, CA: IEEE Computer Society Press, 1998.

JACOBSON, I.; CHRISTERSON, M.; JONSSON, P.; OVERGAARD, G. *Object-oriented software engineering*. Wokingham, UK: Addison-Wesley, 1993.

MARTIN, D.; SOMMERVILLE, I. "Patterns of cooperative interaction: linking ethnomethodology and design." *ACM transactions on computer-human interaction*, v. 11, n. 1, mar. 2004. p. 59-89. doi:10.1145/972648.972651.

RITTEL, H.; WEBBER, M. "Dilemmas in a general theory of planning." *Policy Sciences*, v. 4, 1973. p. 155-169. doi:10.1007/BF01405730.

ROBERTSON, S.; ROBERTSON, J. *Mastering the requirements process*. 3. ed. Boston: Addison-Wesley, 2013.

SOMMERVILLE, I.; RODDEN, T.; SAWYER, P.; BENTLEY, R.; TWIDALE, M. "Integrating ethnography into the requirements engineering process." In: RE' 93. San Diego, CA: IEEE Computer Society Press, 1993. p. 165-173. doi:10.1109/ISRE.1993.324821.

STEVENS, P.; POOLEY, R. *Using UML*: software engineering with objects and components. 2. ed. Harlow, UK: Addison-Wesley, 2006.

SUCHMAN, L. "Office procedures as practical action: models of work and system design." *ACM Transactions on office information systems*, v. 1, n. 3, 1983. p. 320-328. doi:10.1145/357442.357445.

VILLER, S.; SOMMERVILLE, I. "Ethnographically informed analysis for software engineers." *International journal of human-computer studies*, v. 53, n. 1, 2000. p. 169-196. doi:10.1006/ijhc.2000.0370.