CAPÍTULO

4

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26

Engenharia de requisitos

Objetivos

O objetivo deste capítulo é apresentar os requisitos de software e discutir os processos envolvidos na descoberta e documentação desses requisitos. Após a leitura, você:

- compreenderá os conceitos de requisitos de usuário e de sistema e por que eles devem ser escritos de formas diferentes;
- compreenderá as diferenças entre requisitos de software funcionais e não funcionais;
- compreenderá como os requisitos podem ser organizados em um documento de requisitos de software;
- compreenderá as principais atividades de elicitação, análise e validação da engenharia de requisitos e as relações entre essas atividades;
- compreenderá por que o gerenciamento de requisitos é necessário e como ele dá suporte às outras atividades da engenharia de requisitos.

Os requisitos de um sistema são as descrições do que o sistema deve fazer, os serviços que oferece e as restrições a seu funcionamento. Esses requisitos refletem as necessidades dos clientes para um sistema que serve a uma finalidade determinada, como controlar um dispositivo, colocar um pedido ou encontrar informações. O processo de descobrir, analisar, documentar e verificar esses serviços e restrições é chamado engenharia de requisitos (RE, do inglês *requirements engineering)*.

O termo'requisito'não é usado de forma consistente pela indústria de software. Em alguns casos, o requisito é apenas uma declaração abstrata em alto nível de um serviço que o sistema deve oferecer ou uma restrição a um sistema. No outro extremo, é uma definição detalhada e formal de uma função do sistema. Davis (1993) explica por que essas diferenças existem:

> *Se uma empresa pretende fechar um contrato para um projeto de desenvolvimento de software de grande porte, deve definir as necessidades de forma abstrata o suficiente para que a solução para essas necessidades não seja predefinida. Os requisitos precisam ser escritos de modo que vários contratantes possam concorrer pelo contrato e oferecer diferentes maneiras de atenderás necessidades da organização do cliente. Uma vez que o contrato tenha sido adjudicado, o contratante deve escrever para o cliente uma definição mais detalhada do sistema, para que este entenda e possa validar o que o software fará. Ambos os documentos podem ser chamados documentos de requisitos para o sistema.*

Alguns dos problemas que surgem durante o processo de engenharia de requisitos são as falhas em não fazer uma clara

usuário', para expressar os requisitos abstratos de alto nível, e'requisitos de sistema', para expressar a descrição detalhada do que o sistema deve fazer. Requisitos de usuário e requisitos de sistema podem ser definidos como segue:

1. Requisitos de usuário são declarações, em uma linguagem natural com diagramas, de quais serviços o sistema deverá fornecer a seus usuários e as restrições com as quais este deve operar.
2. Requisitos de sistema são descrições mais detalhadas das funções, serviços e restrições operacionais do sistema de software. O documento de requisitos do sistema (às vezes, chamado especificação funcional) deve definir exatamente o que deve ser implementado. Pode ser parte do contrato entre o comprador do sistema e os desenvolvedores de software.

Diferentes níveis de requisitos são úteis, pois eles comunicam informações sobre o sistema para diferentes tipos de leitor. A Figura 4.1 ilustra a distinção entre requisitos de usuário e de sistema. Esse exemplo, de um Sistema de Gerenciamento da Saúde Mental de Pacientes (MHC-PMS, do inglês *Mental Health Care Patient Management System),* mostra como um requisito de usuário pode ser expandido em diversos requisitos de sistemas. Pode-se ver na Figura 4.1 que os requisitos de usuário são mais gerais. Os requisitos de sistema fornecem informações mais específicas sobre os serviços e funções do sistema que devem ser implementados.

Os requisitos precisam ser escritos em diferentes níveis de detalhamento para que diferentes leitores possam usá-los de diversas maneiras. A Figura 4.2 mostra possíveis leitores dos requisitos de usuário e de sistema. Os leitores dos requisitos de usuário não costumam se preocupar com a forma como o sistema será implementado; podem ser gerentes que não estão interessados nos recursos detalhados do sistema. Os leitores dos requisitos de sistema precisam saber mais detalhadamente o que o sistema fará, porque estão interessados em como ele apoiará os processos dos negócios ou porque estão envolvidos na implementação do sistema.

Neste capítulo, apresento uma visão'tradicional'de requisitos e não de requisitos em processos ágeis. Para a maioria dos sistemas de grande porte, ainda é o caso de existir uma fase da engenharia de requisitos claramente identificável antes de se iniciar a implementação do sistema. O resultado é um documento de requisitos, que pode ser parte do contrato de desenvolvimento do sistema. Certamente, haverá mudanças nos requisitos e os requisitos de usuário poderão ser ampliados em requisitos de sistema mais detalhados. No entanto, a abordagem ágil de simultaneamente elicitar os requisitos enquanto o sistema é desenvolvido é raramente utilizada para desenvolvimento de sistemas de grande porte.

Figura 4.1 Requisitos de usuário e de sistema

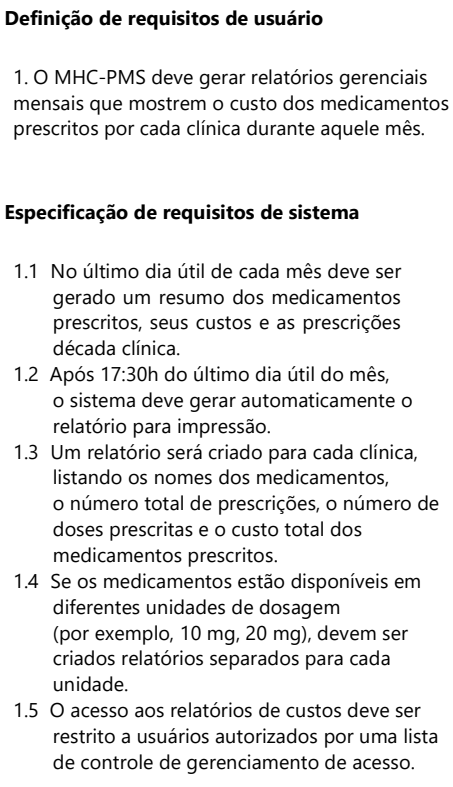
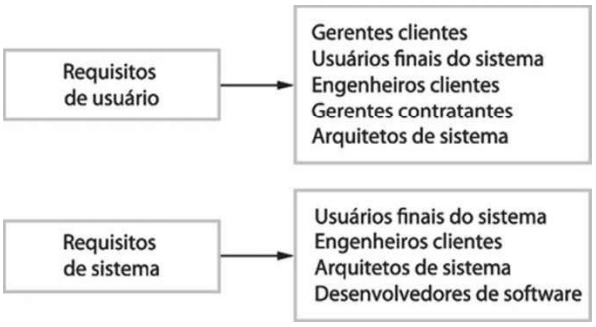


Figura 4.2 Leitores de diferentes tipos de especificação de requisitos



4.1 Requisitos funcionais e não funcionais

Os requisitos de software são frequentemente classificados como requisitos funcionais e requisitos não funcionais:

1. *Requisitos funcionais.* São declarações de serviços que o sistema deve fornecer, de como o sistema deve reagir a entradas específicas e de como o sistema deve se comportar em determinadas situações. Em alguns casos, os requisitos funcionais também podem explicitar o que o sistema não deve fazer.
2. *Requisitos não funcionais.* São restrições aos serviços ou funções oferecidos pelo sistema. Incluem restrições de *timing*, restrições no processo de desenvolvimento e restrições impostas pelas normas. Ao contrário das características individuais ou serviços do sistema, os requisitos não funcionais, muitas vezes, aplicam-se ao sistema como um todo.

Na realidade, a distinção entre diferentes tipos de requisitos não é tão clara como sugerem essas definições simples. Um requisito de usuário relacionado com a proteção, tal como uma declaração de limitação de acesso a 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 recursos de autenticação de usuário no sistema.

Isso mostra que os requisitos não são independentes e que muitas vezes geram ou restringem outros requisitos. Portanto, os requisitos de sistema não apenas especificam os serviços ou as características necessárias ao sistema, mas também a funcionalidade necessária para garantir que esses serviços/características sejam entregues corretamente.

4.1.1 Requisitos funcionais

Os requisitos funcionais de um sistema descrevem o que ele deve fazer. Eles dependem do tipo de software a ser desenvolvido, de quem são seus possíveis usuários e da abordagem geral adotada pela organização ao escrever os requisitos. Quando expressos como requisitos de usuário, os requisitos funcionais são normalmente descritos de forma abstrata, para serem compreendidos pelos usuários do sistema. No entanto, requisitos de sistema funcionais mais específicos descrevem em detalhes as funções do sistema, suas entradas e saídas, exceções etc.

Requisitos funcionais do sistema variam de requisitos gerais, que abrangem o que o sistema deve fazer, até requisitos muito específicos, que refletem os sistemas e as formas de trabalho em uma organização. Por exemplo, aqui estão os exemplos de requisitos funcionais para o sistema MHC-PMS, usados para manter informações sobre os pacientes em tratamento por problemas de saúde mental:

1. Um usuário deve ser capaz de pesquisar as listas de agendamentos para todas as clínicas.
2. O sistema deve gerar a cada dia, para cada clínica, a lista dos pacientes para as consultas daquele dia.
3. Cada membro da equipe que usa o sistema deve ser identificado apenas por seu número de oito dígitos.

Esses requisitos funcionais dos usuários definem os recursos específicos a serem fornecidos pelo sistema. Eles foram retirados do documento de requisitos de usuário e mostram que os requisitos funcionais podem ser escritos em diferentes níveis de detalhamento (contrastar requisitos 1 e 3).

A imprecisão na especificação de requisitos é a causa de muitos problemas da engenharia de software. É compreensível que um desenvolvedor de sistemas interprete um requisito ambíguo de uma maneira que simplifique sua implementação. Muitas vezes, porém, essa não é a preferência do cliente, sendo necessário, então, estabelecer novos requisitos e fazer alterações no sistema. Naturalmente, esse procedimento gera atrasos de entrega e aumenta os custos.

Por exemplo, o primeiro requisito hipotético para o MHC-PMS diz que um usuário deve ser capaz de buscar as listas de agendamentos para todas as clínicas. A justificativa para esse requisito é que os pacientes com problemas de saúde mental por vezes são confusos. Eles podem ter uma consulta em uma clínica, mas se deslocarem até outra. Caso eles tenham uma consulta marcada, eles serão registrados como atendidos, independente da clínica.

O membro da equipe médica que especifica esse requisito pode esperar que'pesquisar'signifique que, dado um nome de paciente, o sistema vai procurar esse nome em todos os agendamentos de todas as clínicas. No entanto, isso não está explícito no requisito. Desenvolvedores do sistema podem interpretar o requisito de maneira diferente e implementar uma pesquisa em que o usuário tenha de escolher uma clínica, e, em seguida, realizar a pesquisa. Isso, obviamente, envolverá mais entradas do usuário e necessitará de mais tempo.

Em princípio, a especificação dos requisitos funcionais de um sistema deve ser completa e consistente. Completude significa que todos os serviços requeridos pelo usuário devem ser definidos. Consistência significa que os requisitos não devem ter definições contraditórias. Na prática, para sistemas grandes e complexos, é praticamente impossível alcançar completude e consistência dos requisitos. Uma razão para isso é que ao elaborar especificações para sistemas complexos é fácil cometer erros e omissões. Outra razão é que em um sistema de grande porte existem muitos *stakeholders*. Um *stakeholderé* uma pessoa ou papel que, de alguma maneira, é afetado pelo sistema. Os *stakeholders* têm necessidades diferentes — e, muitas vezes, inconsistentes. Essas inconsistências podem não ser evidentes em um primeiro momento, quando os requisitos são especificados, e, assim, são incluídas na especificação. Os problemas podem surgir após uma análise mais profunda ou depois de o sistema ter sido entregue ao cliente.

4.1.2 Requisitos não funcionais

Os requisitos não funcionais, como o nome sugere, são requisitos que não estão diretamente relacionados com os serviços específicos oferecidos pelo sistema a seus usuários. Eles podem estar relacionados às propriedades emergentes do sistema, como confiabilidade, tempo de resposta e ocupação de área. Uma alternativa a esse cenário seria os requisitos definirem restrições sobre a implementação do sistema, como as capacidades dos dispositivos de E/S ou as representações de dados usadas nas interfaces com outros sistemas.

Os requisitos não funcionais, como desempenho, proteção ou disponibilidade, normalmente especificam ou restringem as características do sistema como um todo. Requisitos não funcionais são frequentemente mais críticos que requisitos funcionais individuais. Os usuários do sistema podem, geralmente, encontrar maneiras de contornar uma função do sistema que realmente não atenda a suas necessidades. No entanto, deixar de atender a um requisito não funcional pode significar a inutilização de todo o sistema. Por exemplo, se um sistema de aeronaves não cumprir seus requisitos de confiabilidade, não será certificado como um sistema seguro para operar; se um sistema de controle embutido não atender aos requisitos de desempenho, as funções de controle não funcionarão corretamente.

Embora muitas vezes seja possível identificar quais componentes do sistema implementam requisitos funcionais específicos (por exemplo, pode haver componentes de formatação que implementam requisitos de impressão de relatórios), é frequentemente mais difícil relacionar os componentes com os requisitos não funcionais. A implementação desses requisitos pode ser difundida em todo o sistema. Há duas razões para isso:

1. Requisitos não funcionais podem afetar a arquitetura geral de um sistema em vez de apenas componentes individuais. Por exemplo, para assegurar que sejam cumpridos os requisitos de desempenho, será necessário organizar o sistema para minimizar a comunicação entre os componentes.
2. Um único requisito não funcional, tal como um requisito de proteção, pode gerar uma série de requisitos funcionais relacionados que definam os serviços necessários no novo sistema. Além disso, também podem gerar requisitos que restrinjam requisitos existentes.

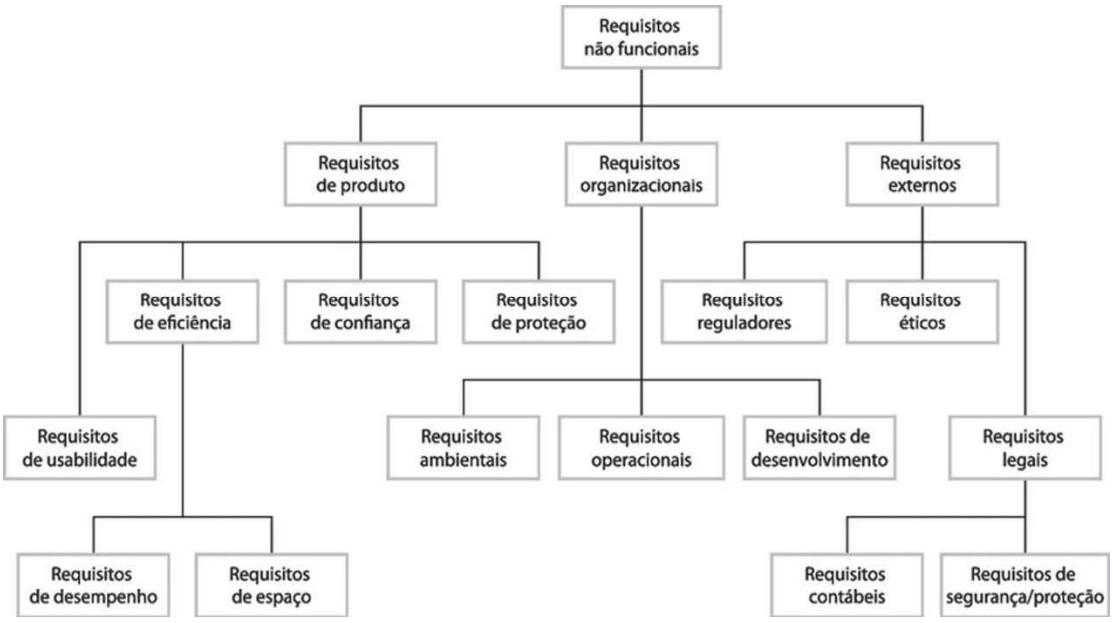
Os requisitos não funcionais surgem por meio das necessidades dos usuários, devido a restrições de orçamento, políticas organizacionais, necessidade de interoperabilidade com outros sistemas de software ou hardware, ou a partir de fatores externos, como regulamentos de segurança ou legislações de privacidade. A Figura 4.3 é uma classificação de requisitos não funcionais. Nesse diagrama você pode ver que os requisitos não funcionais podem ser provenientes das características requeridas para o software (requisitos de produto), da organização que desenvolve o software (requisitos organizacionais) ou de fontes externas:

1. *Requisitos de produto*. Esses requisitos especificam ou restringem o comportamento do software. Exemplos incluem os requisitos de desempenho quanto à rapidez com que o sistema deve executar e quanta memória ele requer, os requisitos de confiabilidade que estabelecem a taxa aceitável de falhas, os requisitos de proteção e os requisitos de usabilidade.
2. *Requisitos organizacionais*. Esses são os requisitos gerais de sistemas derivados das políticas e procedimentos da organização do cliente e do desenvolvedor. Exemplos incluem os requisitos do processo operacional, que definem como o sistema será usado, os requisitos do processo de desenvolvimento que especificam a linguagem de programação, o ambiente de desenvolvimento ou normas de processo a serem usadas, bem como os requisitos ambientais que especificam o ambiente operacional do sistema.
3. *Requisitos externos*. Esse tipo abrange todos os requisitos que derivam de fatores externos ao sistema e seu processo de desenvolvimento. Podem incluir requisitos reguladores, que definem o que deve ser feito para que o sistema seja aprovado para uso, por um regulador, tal como um banco central; requisitos legais, que devem ser seguidos para garantir que o sistema opere dentro da lei; e requisitos éticos, que asseguram que o sistema será aceitável para seus usuários e o público em geral.

O Quadro 4.1 mostra exemplos de requisitos de produto, organizacional e externo retirados do MHC-PMS, cujos requisitos de usuário foram introduzidos na Seção 4.1.1.0 requisito de produto é um requisito de disponibilidade que define quando o sistema deve estar disponível e o tempo diário permitido de seu não funcionamento. Não trata da funcionalidade do MHC-PMS e identifica claramente uma restrição que deve ser considerada pelos projetistas do sistema.

O requisito organizacional especifica como os usuários se autenticam para o sistema. A autoridade de saúde que opera o sistema está migrando para um procedimento de autenticação-padrão para todos os softwares. Neste, em vez de o usuário ter um nome de *login* para se identificar, ele deve passar seu cartão de identificação por um leitor. O requisito externo deriva da necessidade de o sistema estar em conformidade com a legislação de privacidade. A privacidade é obviamente uma questão muito importante nos sistemas de saúde e o requisito especifica claramente que o sistema deverá ser desenvolvido de acordo com uma norma nacional de privacidade.

Figura 4.3 Tipos de requisitos não funcionais



Quadro 4.1 Exemplos de requisitos nâo funcionais no MHC-PMS.

Requisito de produto
0 MHC-PMS deve estar disponível para todas as clínicas durante as horas normais de trabalho (segunda a sexta-feira, 8h30 às 17h30). Períodos de não operação dentro do horário normal de trabalho não podem exceder cinco segundos em um dia.

Requisito organizacional
Usuários do sistema MHC-PMS devem se autenticar com seus cartões de identificação da autoridade da saúde.

Requisito externo
O sistema deve implementar as disposições de privacidade dos pacientes, tal como estabelecido no HStan-03-2006-priv.

Um problema comum com os requisitos não funcionais é que costumam ser propostos pelos usuários ou clientes como metas gerais, em virtude da facilidade de uso, da capacidade do sistema de se recuperar de falhas ou da velocidade das respostas do usuário. Metas estabelecem boas intenções, mas podem causar problemas para os desenvolvedores do sistema, uma vez que deixam margem para interpretação e, consequentemente, para disputas, quando da entrega do sistema. A meta do sistema apresentado a seguir, por exemplo, é típica de como um gerente pode expressar requisitos de usabilidade:

> *O sistema deve ser de fácil uso pelo pessoal médico e deve ser organizado de tal maneira que os erros dos usuários sejam minimizados.*

Reescrevi apenas para mostrar como essa meta poderia ser expressa como um requisito não funcional'testável'. É impossível verificar objetivamente a finalidade do sistema, mas na descrição a seguir é possível incluir pelo menos a instrumentação de software para contar os erros cometidos pelos usuários quando estão testando o sistema.

> *A equipe médica deve ser capaz de usar todas as funções do sistema após quatro horas de treinamento. Após esse treinamento, o número médio de erros cometidos por usuários experientes não deve exceder dois por hora de uso do sistema.*

Sempre que possível, os requisitos não funcionais devem ser escritos quantitativamente, para que possam ser objetivamente testados. A Tabela 4.1 mostra as métricas que você pode usar para especificar as propriedades não funcionais do sistema. Você pode medir essas características quando o sistema está sendo testado para verificar se ele tem cumprido ou não seus requisitos não funcionais.

Na prática, os clientes de um sistema geralmente consideram difícil traduzir suas metas em requisitos mensuráveis. Para algumas metas, como manutenibilidade, não existem métricas que possam ser usadas. Em outros casos, mesmo quando a especificação quantitativa é possível, os clientes podem não ser capazes de relacionar suas necessidades com essas especificações. Eles não entendem o que significa um número definindo a confiabilidade necessária (por exemplo), em termos de sua experiência cotidiana com os sistemas computacionais. Além disso, o custo de verificar requisitos objetivamente não funcionais mensuráveis pode ser muito elevado, e os clientes, que pagam pelo sistema, podem não achar que os custos sejam justificados.

Os requisitos não funcionais frequentemente conflitam e interagem com outros requisitos funcionais ou não funcionais. Por exemplo, o requisito de autenticação no Quadro 4.1 certamente exige a instalação de um leitor de cartão em cada computador conectado ao sistema. No entanto, pode haver outro requisito que solicite acesso móvel ao sistema, pelos laptops dos médicos ou enfermeiras. Estes não são geralmente equipados com leitores de cartão, assim, nessas circunstâncias, algum método de autenticação alternativo deve ser criado.

Na prática, no documento de requisitos, é difícil separar os requisitos funcionais dos não funcionais. Se são apresentados separadamente, os relacionamentos entre eles podem ficar difíceis de serem entendidos. No entanto, os requisitos claramente relacionados com as propriedades emergentes do sistema, como desempenho ou confiabilidade, devem ser explicitamente destacados. Você pode fazer isso colocando-os em uma seção separada do documento de requisitos ou distinguindo-os, de alguma forma, dos outros requisitos de sistema.

Requisitos não funcionais, como confiabilidade, segurança e confidencialidade, são particularmente importantes para sistemas críticos. Escrevo sobre esses requisitos no Capítulo 12, no qual descrevo técnicas próprias para a especificação de requisitos de confiança e de proteção.

Tabela 4.1 Métricas para especificar requisitos não funcionais.

Propriedade	Medida
Velocidade	Transações processadas/segundo Tempo de resposta de usuário/evento Tempo de atualização de tela
Tamanho	Megabytes Número de chips de memória ROM
Facilidade de uso	Tempo de treinamento Número de *frames* de ajuda
Confiabilidade	Tempo médio para falha Probabilidade de indisponibilidade Taxa de ocorrência de falhas Disponibilidade
Robustez	Tempo de reinicio após falha Percentual de eventos que causam falhas Probabilidade de corrupção de dados em caso de falha
Portabilidade	Percentual de declarações dependentes do sistema-alvo Número de sistemas-alvo

4.2 0 documento de requisitos de software

O documento de requisitos de software, às vezes chamado Especificação de Requisitos de Software (SRS — do inglês *Software Requirements Specificotion), é* uma declaração oficial de o que os desenvolvedores do sistema devem implementar. Deve incluir tanto os requisitos de usuário para um sistema quanto uma especificação detalhada dos requisitos de sistema. Em alguns casos, os requisitos de usuário e de sistema são integrados em uma única descrição. Em outros, os requisitos de usuário são definidos em uma introdução à especificação de requisitos de sistema. Se houver um grande número de requisitos, os requisitos detalhados de sistema podem ser apresentados em um documento separado.

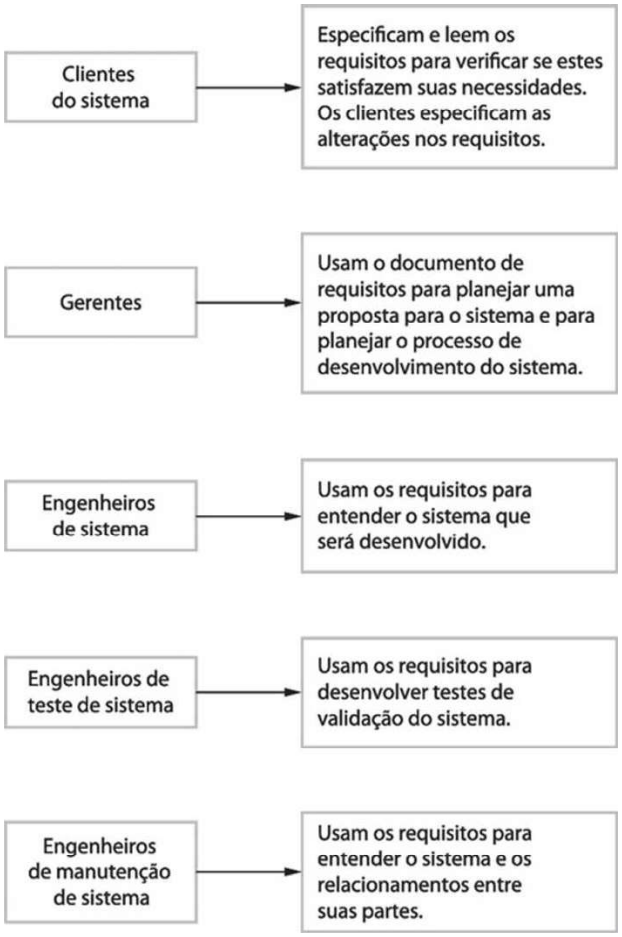
Documentos de requisitos são essenciais quando um contratante externo está desenvolvendo o sistema de software. Entretanto, os métodos ágeis de desenvolvimento argumentam que os requisitos mudam tão rapidamente que um documento de requisitos já está ultrapassado assim que termina de ser escrito. Portanto, o esforço é, em grande parte, desperdiçado. Em vez de um documento formal, abordagens como a Extreme Programming (BECK, 1999) coletam os requisitos de usuário de forma incrementai e escrevem-nos em cartões como estórias de usuário. O usuário então prioriza os requisitos para implementação no próximo incremento do sistema.

Acredito que essa seja uma boa abordagem para os sistemas de negócio em que os requisitos são instáveis. No entanto, penso que ainda é útil escrever um pequeno documento de apoio no qual estejam definidos os requisitos de negócio e de confiança para o sistema; quando o foco está nos requisitos funcionais dos próximos *releases* do sistema, é fácil nos esquecermos dos requisitos que se aplicam ao sistema como um todo.

O documento de requisitos tem um conjunto diversificado de usuários, que vão desde a alta administração da organização que está pagando pelo sistema até os engenheiros responsáveis pelo desenvolvimento do software. A Figura 4.4, tirada do meu livro com Gerald Kotonya sobre engenharia de requisitos (KOTONYA e SOMMERVILLE, 1998) mostra os possíveis usuários do documento e como eles o utilizam.

A diversidade de possíveis usuários é um indicativo de que o documento de requisitos precisa ser um compromisso com a comunicação dos requisitos para os clientes, a definição dos requisitos em detalhes precisos para os desenvolvedores e testadores e a inclusão de informações sobre a possível evolução do sistema. Informações sobre mudanças previstas podem ajudar os projetistas de sistema a evitar decisões de projeto restritivas, além de ajudar os engenheiros de manutenção de sistema que precisam adaptar o sistema aos novos requisitos.

Figura 4.4 Usuários de um documento de engenharia de requisitos.



O nível de detalhes que você deve incluir em um documento de requisitos depende do tipo de sistema em desenvolvimento e o processo usado. Os sistemas críticos precisam ter requisitos detalhados, porque a segurança e a proteção devem ser analisadas em detalhes. Quando o sistema está sendo desenvolvido por uma companhia separada (por exemplo, através de *outsourcing),* as especificações de sistema devem ser detalhadas e precisas. Se um processo interno de desenvolvimento iterativo é usado, o documento de requisitos pode ser muito menos detalhado e quaisquer ambiguidades podem ser resolvidas durante o desenvolvimento do sistema.

A Tabela 4.2 mostra uma possível organização de um documento de requisitos baseada em uma norma IEEE para documentos de requisitos (IEEE, 1998). Essa é uma norma genérica que pode ser adaptada para usos específicos. Nesse caso, eu estendí a norma para incluir informações sobre a evolução prevista do sistema. Essa informação ajuda os engenheiros de manutenção de sistema e permite que os projetistas incluam suporte para futuros recursos do sistema.

Naturalmente, a informação incluída em um documento de requisitos depende do tipo de software a ser desenvolvido e da abordagem de desenvolvimento que está em uso. Se uma abordagem evolutiva é adotada para um produto de software (por exemplo), o documento de requisitos deixará de fora muitos dos capítulos detalhados sugeridos. O foco será sobre a definição de requisitos de usuário e os requisitos não funcionais de alto nível de sistema. Nesse caso, os projetistas e programadores usam seu julgamento para decidir como atender aos requisitos gerais de usuário para o sistema.

No entanto, quando o software é parte de um projeto de um sistema de grande porte que inclui interações entre sistemas de hardware e software, geralmente é necessário definir os requisitos em um alto nível de detalhamento. Isso significa que esses documentos de requisitos podem ser muito longos e devem incluir a maioria ou todos os capítulos mostrados na Tabela 4.2. É particularmente importante incluir uma tabela completa, abrangendo conteúdo e índice de documentos, para que os leitores possam encontrar as informações de que necessitam.

Tabela 4.2 A estrutura de um documento de requisitos.

Capítulo	Descrição
Prefácio	Deve definir os possíveis leitores do documento e descrever seu histórico de versões, incluindo uma justificativa para a criação de uma nova versão e um resumo das mudanças feitas em cada versão.
Introdução	Deve descrever a necessidade para o sistema. Deve descrever brevemente as funções do sistema e explicar como ele vai funcionar com outros sistemas. Também deve descrever como o sistema atende aos objetivos globais de negócio ou estratégicos da organização que encomendou o software.
Glossário	Deve definir os termos técnicos usados no documento. Você não deve fazer suposições sobre a experiência ou o conhecimento do leitor.
Definição de requisitos de usuário	Deve descrever os serviços fornecidos ao usuário. Os requisitos não funcionais de sistema também devem ser descritos nessa seção. Essa descrição pode usar a linguagem natural, diagramas ou outras notações compreensíveis para os clientes. Normas de produto e processos que devem ser seguidos devem ser especificados.
Arquitetura do sistema	Deve apresentar uma visão geral em alto nível da arquitetura do sistema previsto, mostrando a distribuição de funções entre os módulos do sistema. Componentes de arquitetura que são reusados devem ser destacados.
Especificação de requisitos do sistema	Deve descrever em detalhes os requisitos funcionais e não funcionais. Se necessário, também podem ser adicionados mais detalhes aos requisitos não funcionais. Interfaces com outros sistemas podem ser definidas.
Modelos do sistema	Pode incluir modelos gráficos do sistema que mostram os relacionamentos entre os componentes do sistema, o sistema e seu ambiente. Exemplos de possíveis modelos são modelos de objetos, modelos de fluxo de dados ou modelos semânticos de dados.
Evolução do sistema	Deve descrever os pressupostos fundamentais em que o sistema se baseia, bem como quaisquer mudanças previstas, em decorrência da evolução de hardware, de mudanças nas necessidades do usuário etc. Essa seção é útil para projetistas de sistema, pois pode ajudá-los a evitar decisões capazes de restringir possíveis mudanças futuras no sistema.
Apêndices	Deve fornecer informações detalhadas e específicas relacionadas à aplicação em desenvolvimento, além de descrições de hardware e banco de dados, por exemplo. Os requisitos de hardware definem as configurações mínimas ideais para o sistema. Requisitos de banco de dados definem a organização lógica dos dados usados pelo sistema e os relacionamentos entre esses dados.
índice	Vários índices podem ser incluídos no documento. Pode haver, além de um índice alfabético normal, um índice de diagramas, de funções, entre outros pertinentes.

4.3 Especificação de requisitos

A especificação de requisitos é o processo de escrever os requisitos de usuário e de sistema em um documento de requisitos. Idealmente, os requisitos de usuário e de sistema devem ser claros, inequívocos, de fácil compreensão, completos e consistentes. Na prática, isso é difícil de conseguir, pois os *stakeholders* interpretam os requisitos de maneiras diferentes, e, muitas vezes, notam-se conflitos e inconsistências inerentes aos requisitos.

Os requisitos de usuário para um sistema devem descrever os requisitos funcionais e não funcionais de modo que sejam compreensíveis para os usuários do sistema que não tenham conhecimentos técnicos detalhados. Idealmente, eles devem especificar somente o comportamento externo do sistema. O documento de requisitos não deve incluir detalhes da arquitetura ou projeto do sistema. Consequentemente, se você está escrevendo requisitos de usuário, não deve usar o jargão de software, notações estruturadas ou notações formais; você deve escrever os requisitos de usuário em linguagem natural, com tabelas simples, formas e diagramas intuitivos.

Os requisitos de sistema são versões expandidas dos requisitos de usuário, usados por engenheiros de software como ponto de partida para o projeto do sistema. Eles acrescentam detalhes e explicam como os requisitos de usuário devem ser atendidos pelo sistema. Eles podem ser usados como parte do contrato para a implementação do sistema e devem consistir em uma especificação completa e detalhada de todo o sistema.

Os requisitos do sistema devem descrever apenas o comportamento externo do sistema e suas restrições operacionais. Eles não devem se preocupar com a forma como o sistema deve ser projetado ou implementado. No entanto, para atingir o nível de detalhamento necessário para especificar completamente um sistema de software complexo, é praticamente impossível eliminar todas as informações de projeto. Existem várias razões para isso:

1. Você pode precisar projetar uma arquitetura inicial do sistema para ajudar a estruturar a especificação de requisitos. Os requisitos de sistema são organizados de acordo com os diferentes subsistemas que compõem o sistema. Como discuto nos capítulos 6 e 18, essa definição da arquitetura é essencial caso você queira reusar componentes de software na implementação do sistema.
2. Na maioria dos casos, os sistemas devem interoperar com os sistemas existentes, que restringem o projeto e impõem requisitos sobre o novo sistema.
3. O uso de uma arquitetura específica para atender aos requisitos não funcionais (como programação N-version para alcançar a confiabilidade, discutida no Capítulo 13) pode ser necessário. Um regulador externo que precisa certificar que o sistema é seguro pode especificar que um projeto já certificado de arquitetura pode ser usado.

Os requisitos de usuário são quase sempre escritos em linguagem natural e suplementados no documento de requisitos por diagramas apropriados e tabelas. Requisitos de sistema também podem ser escritos em linguagem natural, mas também em outras notações com base em formulários, modelos gráficos ou matemáticos de sistema. ATabela 4.3 resume as notações que poderíam ser usadas para escrever os requisitos de sistema.

Modelos gráficos são mais úteis quando você precisa mostrar como um estado se altera ou quando você precisa descrever uma sequência de ações. Diagramas de sequência e de estado da UML, descritos no Capítulo 5, mostram a sequência de ações que ocorrem em resposta a uma determinada mensagem ou evento. Especificações matemáticas formais são, por vezes, usadas para descrever os requisitos para os sistemas críticos de segurança ou de proteção, mas raramente são usadas em outras circunstâncias. Explico essa abordagem para elaboração de especificações no Capítulo 12.

4.3.1 Especificação em linguagem natural

Desde o início da engenharia de software a linguagem natural tem sido usada para escrever os requisitos para o software. É expressiva, intuitiva e universal.Também é potencialmente vaga, ambígua, e seu significado depende

Tabela 4.3 Formas de escrever uma especificação de requisitos de sistema.

Notação	Descrição
Sentenças em linguagem natural	Os requisitos são escritos em frases numeradas em linguagem natural. Cada frase deve expressar um requisito.
Linguagem natural estruturada	Os requisitos são escritos em linguagem natural em um formulário padrão ou *template*. Cada campo fornece informações sobre um aspecto do requisito.
Linguagem de descrição de projeto	Essa abordagem usa uma linguagem como de programação, mas com características mais abstratas, para especificar os requisitos, definindo um modelo operacional do sistema. Essa abordagem é pouco usada atualmente, embora possa ser útil para as especificações de interface.
Notações gráficas	Para definição dos requisitos funcionais para o sistema são usados modelos gráficos, suplementados por anotações de texto; diagramas de caso de uso e de sequência da UML são comumente usados.
Especificações matemáticas	Essas notações são baseadas em conceitos matemáticos, como máquinas de estado finito ou conjuntos. Embora essas especificações inequívocas possam reduzir a ambiguidade de um documento de requisitos, a maioria dos clientes não entende uma especificação formal. Eles não podem verificar que elas representam o que eles querem e são relutantes em aceitá-las como um contrato de sistema.

do conhecimento do leitor. Como resultado, tem havido muitas propostas de caminhos alternativos para a escrita dos requisitos. No entanto, nenhum destes tem sido amplamente adotado, e a linguagem natural continuará a ser a forma mais usada de especificação de requisitos do software e do sistema.

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

1. Invente um formato-padrão e garanta que todas as definições de requisitos aderem a esse formato. A padronização do formato torna menos prováveis as omissões e mais fácil a verificação dos requisitos. O formato que eu uso expressa o requisito em uma única frase. Eu associo uma declaração com a justificativa para cada requisito de usuário para explicar por que o requisito foi proposto. A justificativa também pode incluir informações sobre quem propôs o requisito (a fonte do requisito), para saber quem consultar caso o requisito tenha de ser mudado.
2. Use uma linguagem consistente para distinguir entre os requisitos obrigatórios e os desejáveis. Os obrigatórios são requisitos aos quais o sistema tem de dar suporte e geralmente são escritos usando-se deve' Requisitos desejáveis não são essenciais e são escritos usando-se'pode'.
3. Use uma forma de destacar as partes fundamentais do requisito (negrito, itálico ou cores).
4. Não assuma que os leitores compreendem a linguagem técnica da engenharia de software. Frequentemente, palavras como'arquitetura'e'módulo'são mal interpretadas. Você deve, portanto, evitar o uso de jargões, siglas e acrônimos.
5. Sempre que possível, tente associar uma lógica a cada um dos requisitos de usuário. Essa justificativa deve explicar por que o requisito foi incluído, e é particularmente útil quando os requisitos são alterados, uma vez que pode ajudar a decidir sobre quais mudanças seriam indesejáveis.

O Quadro 4.2 ilustra como essas diretrizes podem ser usadas. Inclui dois requisitos para o software embutido para a bomba automática de insulina, citada no Capítulo 1. Você pode baixar a especificação de requisitos para a bomba de insulina completa das páginas do livro na Internet.

4.3.2 Especificações estruturadas

A linguagem natural estruturada é uma forma de escrever os requisitos do sistema na qual a liberdade do escritor dos requisitos é limitada e todos os requisitos são escritos em uma forma-padrão. Essa abordagem mantém grande parte da expressividade e compreensão da linguagem natural, mas garante certa uniformidade imposta sobre a especificação. Notações de linguagem estruturada usam *templates* para especificar os requisitos de sistema. A especificação pode usar construções de linguagem de programação para mostrar alternativas e iteração; além disso, pode destacar elementos-chave pelo uso de sombreamento ou fontes diferentes.

Os Robertson (ROBERTSON e ROBERTSON, 1999), em seu livro sobre o método VOLERE de engenharia de requisitos, recomendam que os requisitos de usuário sejam inicialmente escritos em cartões, um requisito por cartão. Eles sugerem um número de campos em cada cartão, algo como a lógica dos requisitos, as dependências de outros requisitos, a origem dos requisitos, materiais de apoio, e assim por diante. Essa é uma abordagem semelhante à do exemplo de uma especificação estruturada do Quadro 4.3.

Para usar uma abordagem estruturada para especificação de requisitos de sistema, você pode definir um ou mais *templates* para representar esses requisitos como formulários estruturados. A especificação pode ser estruturada em torno dos objetos manipulados pelo sistema, das funções desempenhadas pelo sistema, ou pelos eventos processados pelo sistema. Um exemplo de uma especificação baseada em formulários, nesse caso, que define a

Quadro 4.2 Exemplo de requisitos para o sistema de software de bomba de insulina.

3.2 0 sistema deve medir o açúcar no sangue e fornecer insulina, se necessário, a cada dez minutos. *(Mudanças de açúcar no sangue são relativamente lentas, portanto, medições mais frequentes são desnecessárias; medições menos frequentes podem levar a níveis de açúcar desnecessariamente elevados.)*

3.6 0 sistema deve, a cada minuto, executar uma rotina de autoteste com as condições a serem testadas e as ações associadas definidas na Quadro 4.3 *{A rotina de autoteste pode descobrir problemas de hardware e software e pode alertar o usuário para a impossibilidade de operar normalmente.)*

Quadro 4.3 Uma especificação estruturada de um requisito para uma bomba de insulina.

Bomba de insulina/Software de controle/SRS/3.3.2

Função	Calcula doses de insulina: nível seguro de açúcar.
Descrição	Calcula a dose de insulina a ser fornecida quando o nível de açúcar está na zona de segurança entre três e
Entradas	sete unidades.
Fonte	Leitura atual de açúcar (r2), duas leituras anteriores (rO e r1).
Saídas	Leitura atual da taxa de açúcar pelo sensor. Outras leituras da memória.
Destino	CompDose — a dose de insulina a ser fornecida.
Ação	*Loop* principal de controle.
Requisitos	CompDose é zero se o nível de açúcar está estável ou em queda ou se o nível está aumentando, mas a taxa de aumento está diminuindo. Se o nível está aumentando e a taxa de aumento está aumentando, então CompDose é calculado dividindo-se a diferença entre o nível atual de açúcar e o nível anterior por quatro e arredondando-se o resultado.
Pré-condição	Se o resultado é arredondado para zero, então CompDose é definida como a dose mínima que pode ser fornecida por r2.
Pós-condições	rO é substituída por r1 e r1 é substituída
Efeitos colaterais	

forma de calcular a dose de insulina a ser fornecida quando o açúcar no sangue está dentro de uma faixa de segurança, é mostrado no Quadro 4.3

Quando um formulário-padrão é usado para especificar requisitos funcionais, as seguintes informações devem ser incluídas:

1. A descrição da função ou entidade a ser especificada.
2. Uma descrição de suas entradas e de onde elas vieram.
3. Uma descrição de suas saídas e para onde elas irão.
4. Informações sobre a informação necessária para o processamento ou outras entidades usadas no sistema (a parte *'requires')*.
5. Uma descrição da ação a ser tomada.
6. Se uma abordagem funcional é usada, uma pré-condição define o que deve ser verdade antes que a função seja chamada, e é chamada uma pós-condição, especificando o que é verdade depois da função.
7. Uma descrição dos efeitos colaterais da operação (caso haja algum).

Usar as especificações estruturadas elimina alguns dos problemas de especificação em linguagem natural. Reduz-se a variabilidade na especificação, e os requisitos são organizados de forma mais eficaz. No entanto, algumas vezes ainda é difícil escrever os requisitos de forma clara e inequívoca, especialmente quando processamentos complexos (como, por exemplo, calcular a dose de insulina) devem ser especificados.

Para resolver esse problema, você pode adicionar informações extras aos requisitos em linguagem natural, usando tabelas ou modelos gráficos do sistema, por exemplo. Estes podem mostrar como os cálculos são executados, como o sistema muda de estado, como os usuários interagem com o sistema e como sequências de ações são executadas.

As tabelas são particularmente úteis quando há um número de situações alternativas possíveis e é necessário descrever as ações a serem tomadas para cada uma delas. A bomba de insulina baseia seus cálculos na necessidade de insulina sobre a taxa de variação dos níveis de açúcar no sangue. As taxas de variação são calculadas usando as leituras atuais e anteriores. ATabela 4.4 é uma descrição tabular de como a taxa de variação de açúcar no sangue é usada para calcular a quantidade de insulina a ser fornecida.

Tabela 4.4 Especificação tabular de processamento para uma bomba de insulina.

Condição	Ação
Nível de açúcar diminuindo (r2 < rl)	CompDose = 0
Nível de açúcar estável (r2 = rl)	CompDose = 0
Nível de açúcar aumentando e a taxa de aumento decrescente [(r2 - r1) < (rl - r0)]	CompDose = 0
Nível de açúcar aumentando e a taxa de aumento estável ou crescente [(r2-r1)^(r1 — r0)]	CompDose = arredondar [(r2 - r1) / **4)].** Se o resultado arredondado = 0, então CompDose = MinimumDose

4.4 Processos de engenharia de requisitos

Como discutido no Capítulo 2, os processos de engenharia de requisitos podem incluir quatro atividades de alto nível. Elas visam avaliar se o sistema é útil para a empresa (estudo de viabilidade), descobrindo requisitos felicitação e análise), convertendo-os em alguma forma-padrão (especificação), e verificar se os requisitos realmente definem o sistema que o cliente quer (validação). Mostrei essas atividades como processos sequenciais na Figura 2.6. No entanto, na prática, a engenharia de requisitos é um processo iterativo em que as atividades são intercaladas.

A Figura 4.4 mostra esta intercalação. As atividades são organizadas em torno de uma espiral, como um processo
iterativo, sendo a saída um documento de requisitos de sistema. A quantidade de tempo e esforço dedicados a cada atividade em cada iteração depende do estágio do processo como um todo e do tipo de sistema que está sendo desenvolvido. No início do processo, o esforço maior será a compreensão dos requisitos de negócio e não funcionais em
alto nível, bem como dos requisitos de usuário para o sistema. Mais tarde no processo, nos anéis externos da espiral, o esforço maior será dedicado a elicitar e compreender os requisitos de sistema em detalhes.

Esse modelo espiral acomoda abordagens em que os requisitos são desenvolvidos em diferentes níveis de detalhamento. O número de iterações em torno da espiral pode variar; assim, a espiral pode acabar depois da definição de alguns ou de todos os requisitos de usuário. No lugar de prototipação, o desenvolvimento ágil pode ser usado para que os requisitos e a implementação do sistema sejam desenvolvidos em conjunto.

Algumas pessoas consideram a engenharia de requisitos o processo de aplicação de um método de análise estruturada, como a análise orientada a objetos (LARMAN, 2002). Trata-se de analisar o sistema e desenvolver um conjunto de modelos gráficos de sistema, como modelos de casos de uso, que, então, servem como uma especificação do sistema. O conjunto de modelos descreve o comportamento do sistema e é anotado com informações adicionais, descrevendo, por exemplo, o desempenho ou a confiabilidade requerida do sistema.

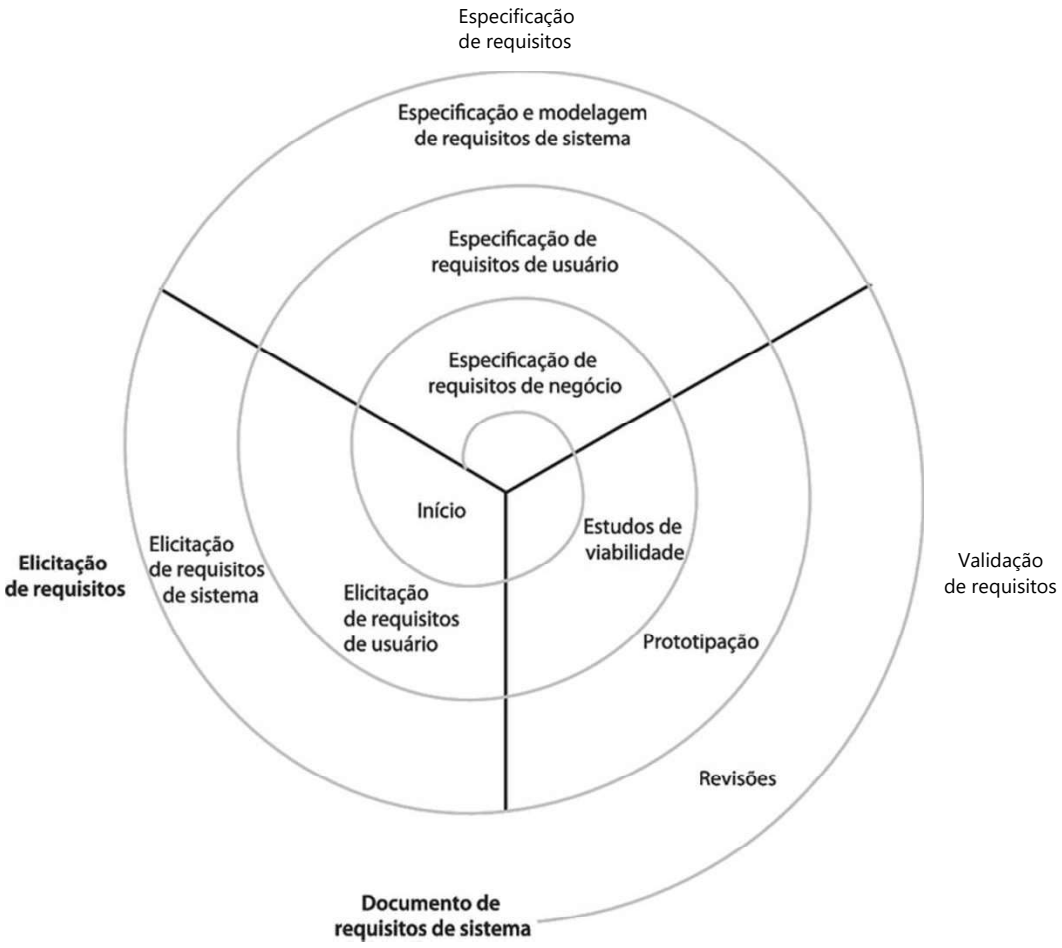
Embora os métodos estruturados tenham um papel a desempenhar no processo de engenharia de requisitos, existe muito mais para a engenharia de requisitos do que o que é coberto por esses métodos. Elicitação de requisitos, em particular, é uma atividade centrada em pessoas, e as pessoas não gostam de restrições impostas por modelos rígidos de sistema.

Em praticamente todos os sistemas os requisitos mudam. As pessoas envolvidas desenvolvem uma melhor compreensão do que querem do software, a organização que compra o sistema também muda, modificações são feitas no hardware, no software e no ambiente organizacional do sistema. O processo de gerenciamento desses requisitos em constante mudança é chamado gerenciamento de requisitos, sobre o qual eu escrevo na

4.5 Elicitação e análise de requisitos

Após um estudo inicial de viabilidade, o próximo estágio do processo de engenharia de requisitos é a elicitação e análise de requisitos. Nessa atividade, os engenheiros de software trabalham com clientes e usuários finais do sistema para obter informações sobre o domínio da aplicação, os serviços que o sistema deve oferecer, o desempenho do sistema, restrições de hardware e assim por diante.

Figura 4.5 Uma visão em espiral do processo de engenharia de requisitos.



A elicitação e análise de requisitos podem envolver diversos tipos de pessoas em uma organização. Um *stakeholderôo* sistema é quem tem alguma influência direta ou indireta sobre os requisitos do sistema. Os *stakeholders* incluem os usuários finais que irão interagir com o sistema e qualquer outra pessoa em uma organização que será afetada por ele. Outros *stakeholders* do sistema podem ser os engenheiros que estão desenvolvendo ou mantendo outros sistemas relacionados a esse, como também gerentes de negócios, especialistas de domínio e representantes sindicais.

Um modelo do processo de elicitação e análise é mostrado na Figura 4.6. Cada organização terá sua própria versão ou instância desse modelo geral, dependendo de fatores locais, como a *expertise* do pessoal, o tipo de sistema a ser desenvolvido, as normas usadas etc.

As atividades do processo são:

1. *Descoberto de requisitos.* Essa é a atividade de interação com os *stakeholders* do sistema para descobrir seus requisitos. Os requisitos de domínio dos *stakeholders* e da documentação também são descobertos durante essa atividade. Existem várias técnicas complementares que podem ser usadas para descoberta de requisitos, que discuto mais adiante.
2. *Classificação e organização de requisitos.* Essa atividade toma a coleção de requisitos não estruturados, agrupa requisitos relacionados e os organiza em grupos coerentes. A forma mais comum de agrupar os requisitos é o uso de um modelo de arquitetura do sistema para identificar subsistemas e associar requisitos a cada subsistema. Na prática, a engenharia de requisitos e projeto da arquitetura não podem ser atividades completamente separadas.
3. *Priorização e negociação de requisitos.* Inevitavelmente, quando os vários *stakeholders* estão envolvidos, os requisitos entram em conflito. Essa atividade está relacionada com a priorização de requisitos e em encontrar e

Figura 4.6 O processo de elicitação e análise de requisitos.

resolver os conflitos por meio da negociação de requisitos. Normalmente, os *stakeholders* precisam se encontrar para resolver as diferenças e chegar a um acordo sobre os requisitos.

4. *Especificação de requisitos*. Os requisitos são documentados e inseridos no próximo ciclo da espiral. Documentos formais ou informais de requisitos podem ser produzidos, como discutido na Seção 4.3.

A Figura 4.5 mostra que a elicitação e análise de requisitos é um processo iterativo, com *feedback* contínuo de cada atividade para as outras atividades. O ciclo do processo começa com a descoberta de requisitos e termina com sua documentação. O entendimento do analista de requisitos melhora a cada rodada do ciclo. Quando se completa o documento de requisitos, o ciclo termina.

Elicitar e compreender os requisitos dos *stakeholders* do sistema é um processo difícil por várias razões:

1. Exceto em termos gerais, os *stakeholders* costumam não saber o que querem de um sistema computacional; eles podem achar difícil articular o que querem que o sistema faça, e, como não sabem o que é viável e o que não é, podem fazer exigências inviáveis.
2. Naturalmente, os *stakeholders* expressam requisitos em seus próprios termos e com o conhecimento implícito de seu próprio trabalho. Engenheiros de requisitos, sem experiência no domínio do cliente, podem não entender esses requisitos.
3. Diferentes *stakeholders* têm requisitos diferentes e podem expressar essas diferenças de várias maneiras. Engenheiros de requisitos precisam descobrir todas as potenciais fontes de requisitos e descobrir as semelhanças e conflitos.
4. Fatores políticos podem influenciar os requisitos de um sistema. Os gerentes podem exigir requisitos específicos, porque estes lhes permitirão aumentar sua influência na organização.
5. O ambiente econômico e empresarial no qual a análise ocorre é dinâmico. É inevitável que ocorram mudanças durante o processo de análise. A importância dos requisitos específicos pode mudar. Novos requisitos podem surgir a partir de novos *stakeholders* que não foram inicialmente consultados.

Inevitavelmente, os diferentes *stakeholders* têm opiniões diferentes sobre a importância e prioridade dos requisitos e, por vezes, essas opiniões são conflitantes. Durante o processo, você deve organizar as negociações regulares dos *stakeholders*, de modo que os compromissos possam ser cumpridos. É completamente impossível satisfazer a todos os *stakeholders*, mas, se alguns deles perceberem que suas opiniões não foram devidamente consideradas, poderão, deliberadamente, tentar arruinar o processo de RE.

No estágio de especificação de requisitos, aqueles que foram elicitados até esse momento são documentados de forma a ajudar na descoberta de novos requisitos. Nesse estágio, uma versão inicial do documento de requisitos do sistema pode ser produzida com seções faltantes e requisitos incompletos. Como alternativa, os requisitos podem ser documentados de uma forma completamente diferente (por exemplo, em uma planilha ou em cartões). Escrever os requisitos em cartões pode ser muito eficaz, pois são fáceis para os *stakeholders* lidarem, mudarem e organizarem.

í&é4*5*1 Descoberta de requisitos

A descoberta de requisitos (às vezes, chamada elicitaçâo de requisitos) é o processo de reunir informações sobre o sistema requerido e os sistemas existentes e separar dessas informações os requisitos de usuário e de sistema. Fontes de informação durante a fase de descoberta de requisitos incluem documentação, *stokeholders* do sistema e especificações de sistemas similares. Você interage com os *stakeholders* por meio da observação e de entrevistas e pode usar cenários e protótipos para ajudar os *stokeholders* a compreenderem o que o sistema vai ser.

Os *stakeholders* variam desde os usuários finais, passando pelos gerentes do sistema até *stokeholders* externos, como reguladores, que certificam a aceitabilidade do sistema. Por exemplo, os *stokeholders* do sistema de informação da saúde mental de pacientes incluem:

1. Os pacientes cujas informações estão registradas no sistema.
2. Os médicos responsáveis pela avaliação e tratamento dos pacientes.
3. Os enfermeiros que, alinhados com os médicos, coordenam as consultas e administram tratamentos.
4. As (os) recepcionistas dos médicos, que gerenciam as consultas dos pacientes.
5. A equipe de TI, responsável pela instalação e manutenção do sistema.
6. Um gerente de ética médica, que deve garantir que o sistema atenda às diretrizes éticas atuais no atendimento ao paciente.
7. Gerentes de saúde, que obtêm informações de gerenciamento a partir do sistema.
8. A equipe de registros médicos, responsável por garantir que as informações do sistema sejam mantidas e preservadas, e que os procedimentos de manutenção dos registros sejam devidamente implementados.

Além dos *stokeholders* do sistema, já vimos que os requisitos também podem vir a partir do domínio da aplicação e de outros sistemas que interagem com o sistema especificado. Durante o processo de elicitaçâo de requisitos, todos esses devem ser considerados.

Essas diferentes fontes de requisitos (.*stokeholders*, domínio, sistemas) podem ser representadas como pontos de vista do sistema, com cada ponto de vista mostrando um subconjunto dos requisitos para o sistema. Pontos de vista diferentes sobre um problema percebem o problema de maneiras diferentes. No entanto, suas perspectivas não são completamente independentes; em geral, elas se sobrepõem e, dessa forma, apresentam requisitos comuns. Você pode usar esses pontos de vista para estruturar a descoberta e a documentação dos requisitos do sistema.

4,5,2 Entrevistas

Entrevistas formais ou informais com os *stokeholders* do sistema são parte da maioria dos processos de engenharia de requisitos. Nessas entrevistas, a equipe de engenharia de requisitos questiona os *stokeholders* sobre o sistema que usam no momento e sobre o sistema que será desenvolvido. Requisitos surgem a partir das respostas a essas perguntas. As entrevistas podem ser de dois tipos:

1. Entrevistas fechadas, em que o *stakeholder* responde a um conjunto predefinido de perguntas.
2. Entrevistas abertas, em que não existe uma agenda predefinida. A equipe de engenharia de requisitos explora uma série de questões com os *stokeholders* do sistema e, assim, desenvolve uma melhor compreensão de suas necessidades.

Na prática, as entrevistas com os *stokeholders* costumam ser uma mistura de ambos os tipos. Você poderá ter de obter a resposta a determinadas questões, mas é comum que estas levem a outras, discutidas de forma menos estruturada. Discussões totalmente abertas raramente funcionam bem. Você geralmente tem de fazer algumas perguntas para começar e manter a entrevista centrada no sistema que será desenvolvido.

Entrevistas são boas para obter uma compreensão global sobre o que os *stokeholders* fazem, como eles podem interagir com o novo sistema e as dificuldades que eles enfrentam com os sistemas atuais. As pessoas gostam de falar sobre seus trabalhos e geralmente ficam felizes de se envolver em entrevistas. No entanto, as entrevistas não são tão úteis na compreensão dos requisitos do domínio da aplicação.

Pode ser difícil elicitar conhecimento do domínio por meio de entrevistas por duas razões:

1. Todos os especialistas em aplicações usam terminologias e jargões específicos para um domínio. Para eles, é impossível discutir os requisitos de domínio essa terminologia. Eles normalmente usam a terminologia de forma precisa e sutil, o que dificulta a compreensão dos engenheiros de requisitos.
2. O conhecimento de domínio é tão familiar aos *stokeholders* que eles têm dificuldade de explicá-lo, ou pensam que é tão fundamental que não vale a pena mencionar. Por exemplo, para um bibliotecário, não é necessário dizer que todas as aquisições são catalogadas antes de serem adicionadas à biblioteca. No entanto, isso pode não ser óbvio para o entrevistador, e acaba não sendo levado em conta nos requisitos.

Entrevistas também não são uma técnica eficaz para a elicitação do conhecimento sobre os requisitos e restrições organizacionais, porque, entre as diferentes pessoas da organização, existem sutis relações de poder. As estruturas organizacionais publicadas raramente correspondem à realidade do processo de tomada de decisão em uma organização, mas os entrevistados podem preferir não revelar a estrutura real, e sim a estrutura teórica, para um estranho. Em geral, a maioria das pessoas é relutante em discutir questões políticas e organizacionais que podem afetar os requisitos.

Entrevistadores eficazes têm duas características:

1. Eles estão abertos a novas idéias, evitam idéias preconcebidas sobre os requisitos e estão dispostos a ouvir os *stakeholders*. Mesmo que o *stakeholder* apresente requisitos-surpresa, eles estão dispostos a mudar de ideia sobre o sistema.

2. Eles estimulam o entrevistado a participar de discussões com uma questão-trampolim, uma proposta de requisitos ou trabalhando em conjunto em um protótipo do sistema. É improvável que dizer às pessoas "diga-me o que quiser" resulte em informações úteis. É muito mais fácil falar em um contexto definido do que em termos gerais.

Informações recolhidas em entrevistas suplementam outras informações sobre o sistema, advindas de documentos que descrevem processos de negócios ou sistemas existentes, observações do usuário etc. Em alguns casos, além da informação contida nos documentos do sistema, as entrevistas podem ser a única fonte de informação sobre os requisitos do sistema. No entanto, a entrevista por si só pode deixar escapar informações essenciais; por isso, deve ser usada em conjunto com outras técnicas de elicitação de requisitos.

4.5.3 Cenários

As pessoas geralmente acham mais fácil se relacionar com exemplos da vida real do que com descrições abstratas. Elas podem compreender e criticar um cenário de como elas podem interagir com um sistema de software. Engenheiros de requisitos podem usar a informação obtida a partir deste debate para formular os requisitos do sistema atual.

Os cenários podem ser particularmente úteis para adicionar detalhes a uma descrição geral de requisitos.Trata-se de descrições de exemplos de sessões de interação. Cada cenário geralmente cobre um pequeno número de interações possíveis. Diferentes cenários são desenvolvidos e oferecem diversos tipos de informação em variados níveis de detalhamento sobre o sistema. As estórias usadas em Extreme Programming, discutidas no Capítulo 3, são um tipo de cenário de requisitos.

Um cenário começa com um esboço da interação. Durante o processo de elicitação, são adicionados detalhes ao esboço, para criar uma descrição completa dessa interação. Em sua forma mais geral, um cenário pode incluir:

1. Uma descrição do que o sistema e os usuários esperam quando o cenário se iniciar.

2. Uma descrição do fluxo normal de eventos no cenário.

3. Uma descrição do que pode dar errado e como isso é tratado.

4. Informações sobre outras atividades que podem acontecer ao mesmo tempo.

5. Uma descrição do estado do sistema quando o cenário acaba.

A elicitação baseada em cenários envolve o trabalho com os *stakeholders* para identificar cenários e capturar detalhes que serão incluídos nesses cenários. Os cenários podem ser escritos como texto, suplementados por diagramas, telas etc. Outra possibilidade é uma abordagem mais estruturada, em que cenários de eventos ou casos de uso podem ser usados.

Como exemplo de um cenário de texto simples, considere como o MHC-PMS pode ser usado para introduzir dados de um novo paciente (Quadro 4.4). Quando um novo paciente vai a uma clínica, um novo registro é criado

Quadro 4.4 Cenário para a coleta do histórico médico em MHC-PMS.

Suposição inicial:
0 paciente é atendido em uma clínica médica por uma recepcionista; ela gera um registro no sistema e coleta suas informações pessoais (nome, endereço, idade etc.). Uma enfermeira é conectada ao sistema e coleta o histórico médico do paciente.

Normal:
A enfermeira busca o paciente pelo sobrenome. Se houver mais de um paciente com o mesmo sobrenome, o nome e a data de nascimento são usados para identificar o paciente.
A enfermeira escolhe a opção do menu para adicionar o histórico médico.
A enfermeira segue, então, uma série de *prompts* do sistema para inserir informações sobre consultas em outros locais, os problemas de saúde mental (entrada de texto livre), condições médicas (enfermeira seleciona condições do menu), medicação atual (selecionado no menu), alergias (texto livre) e informações da vida doméstica (formulário).

O que pode dar errado:
O prontuário do paciente não existe ou não pôde ser encontrado. A enfermeira deve criar um novo registro e registrar as informações pessoais.
As condições do paciente ou a medicação em uso não estão inscritas no menu. A enfermeira deve escolher a opção 'outros'e inserir texto livre com descrição da condição/medicação.
O paciente não pode/não fornecerá informações sobre seu histórico médico. A enfermeira deve inserir um texto livre registrando a incapacidade/ relutância do paciente em fornecer as informações. O sistema deve imprimir o formulário-padrão de exclusão afirmando que a falta de informação pode significar que o tratamento será limitado ou postergado. Este deverá ser assinado e entregue ao paciente.

Outras atividades:
Enquanto a informação está sendo inserida, o registro pode ser consultado, mas não editado por outros agentes.

Estado do sistema na conclusão:
O usuário está conectado. O prontuário do paciente, incluindo seu histórico médico, é inserido no banco de dados e um registro é adicionado ao *log* do sistema, mostrando o tempo de início e fim da sessão e a enfermeira envolvida.

por uma recepcionista, e suas informações pessoais (nome, idade etc.) são acrescentadas nessa ficha. Em seguida, uma enfermeira entrevista o paciente e coleta seu histórico médico. Então, o paciente faz uma consulta inicial com o médico, que faz um diagnóstico e, se for o caso, recomenda um tratamento. O cenário mostra o que acontece quando o histórico médico é coletado.

4.5.4 Casos de uso

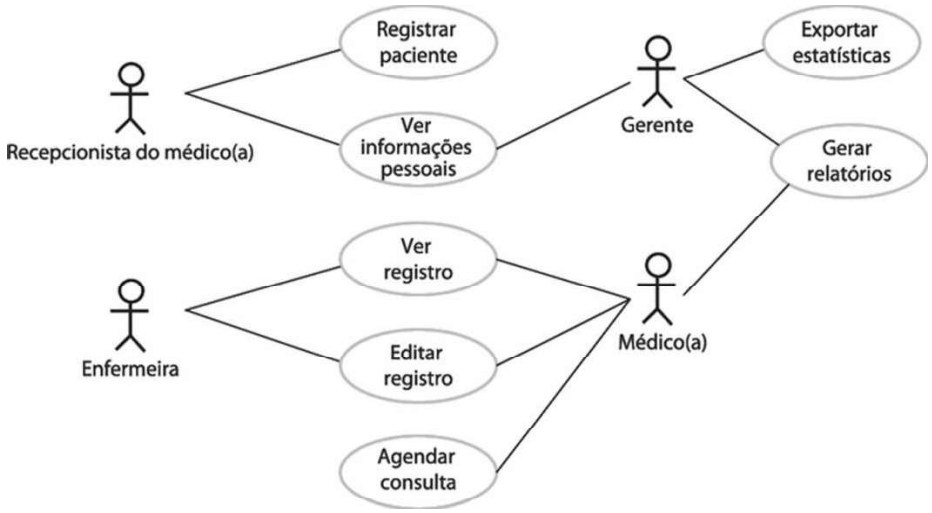
Os casos de uso são uma técnica de descoberta de requisitos introduzida inicialmente no método Objectory (JACOBSON et al., 1993). Eles já se tornaram uma característica fundamental da linguagem de modelagem unificada (UML — do inglês *unified modeling language).* Em sua forma mais simples, um caso de uso identifica os atores envolvidos em uma interação e dá nome ao tipo de interação. Essa é, então, suplementada por informações adicionais que descrevem a interação com o sistema. A informação adicional pode ser uma descrição textual ou um ou mais modelos gráficos, como diagrama de sequência ou de estados da UML.

Os casos de uso são documentados por um diagrama de casos de uso de alto nível. O conjunto de casos de uso representa todas as possíveis interações que serão descritas nos requisitos de sistema. Atores, que podem ser pessoas ou outros sistemas, são representados como figuras'palito'. Cada classe de interação é representada por uma elipse. Linhas fazem a ligação entre os atores e a interação. Opcionalmente, pontas de flechas podem ser adicionadas às linhas para mostrar como a interação se inicia. Essa situação é ilustrada na Figura 4.6, que mostra alguns dos casos de uso para o sistema de informações de pacientes.

Não há distinção entre cenários e casos de uso que seja simples e rápida. Algumas pessoas consideram cada caso de uso um cenário único; outros, como sugerido por Stevens e Pooley (2006), encapsulam um conjunto de cenários em um único caso de uso. Cada cenário é um segmento através do caso de uso. Portanto, seria um cenário para a interação normal além de cenários para cada possível exceção. Você pode, na prática, usá-los de qualquer forma.

Os casos de uso identificam as interações individuais entre o sistema e seus usuários ou outros sistemas. Cada caso de uso deve ser documentado com uma descrição textual. Esta, por sua vez, pode ser ligada a outros modelos

Figura 4.6 Casos de uso para o MHC-PMS.



UML que desenvolverão o cenário com mais detalhes. Por exemplo, uma breve descrição do caso de uso Agendar consulta, representado na Figura 4.6, pode ser:

> *Agendar a consulta permite que dois ou mais médicos de consultórios diferentes possam ler o mesmo registro ao mesmo tempo. Um médico deve escolher, em um menu de lista de médicos on-line, as pessoas envolvidas. 0 prontuário do paciente é então exibido em suas telas, mas apenas o primeiro médico pode editar o registro. Além disso, uma janela de mensagens de texto é criada para ajudar a coordenar as ações. Supõe-se que uma conferência telefônica para comunicação por voz será estabelecida separadamente.*

Cenários e casos de uso são técnicas eficazes para elicitar requisitos dos *stakeholders* que vão interagir diretamente com o sistema. Cada tipo de interação pode ser representado como um caso de uso. No entanto, devido a seu foco nas interações com o sistema, eles não são tão eficazes para elicitar restrições ou requisitos de negócios e não funcionais em alto nível ou para descobrir requisitos de domínio.

A UML é, de fato, um padrão para a modelagem orientada a objetos, e assim, casos de uso e elicitação baseada em casos de uso são amplamente usados para a elicitação de requisitos. Mais adiante, no Capítulo 5, discuto casos de uso e mostro como eles são usados, juntamente com outros modelos, para documentar um projeto de sistema.

l&t⁴*5*5 Etnografia

Os sistemas de software não existem isoladamente. Eles são usados em um contexto social e organizacional, e requisitos de software do sistema podem ser derivados ou restringidos desse contexto. Geralmente, satisfazer a esses requisitos sociais e organizacionais é crítico para o sucesso do sistema. Uma razão pela qual muitos sistemas de software são entregues, mas nunca são usados, é que seus requisitos não levam devidamente em conta a forma como o contexto social e organizacional afeta o funcionamento prático do sistema.

Etnografia é uma técnica de observação que pode ser usada para compreender os processos operacionais e ajudar a extrair os requisitos de apoio para esses processos. Um analista faz uma imersão no ambiente de trabalho em que o sistema será usado. O trabalho do dia a dia é observado e são feitas anotações sobre as tarefas reais em que os participantes estão envolvidos. O valor da etnografia é que ela ajuda a descobrir requisitos implícitos do sistema que refletem as formas reais com que as pessoas trabalham, em vez de refletir processos formais definidos pela organização.

Frequentemente, as pessoas acham muito difícil expressar os detalhes de seu trabalho, pois isso é secundário para elas. Elas entendem o próprio trabalho, mas não compreendem sua relação com outros trabalhos na organização. Fatores sociais e organizacionais que afetam o trabalho, mas que não são óbvios para os indivíduos, podem ficar claros apenas quando analisados por um observador imparcial. Por exemplo, um grupo de trabalho pode se auto-organizar de maneira que os membros conheçam mutuamente seus trabalhos e, em caso de ausência de algum deles, possam dividir as tarefas. Essa informação pode não ser mencionada durante uma entrevista, pois o grupo não a percebe como parte integrante de seu trabalho.

Suchman (1987) foi pioneira no uso da etnografia na análise do trabalho de escritório. Ela constatou que as práticas de trabalho eram muito mais ricas, mais complexas e mais dinâmicas do que os modelos simples assumidos pelos sistemas de automação de escritório. A diferença entre o trabalho presumido e o real foi a razão mais importante por que esses sistemas de escritório não causaram efeitos significativos sobre a produtividade. Crabtree (2003) discute uma ampla gama de estudos desde então e descreve, em geral, o uso da etnografia em projeto de sistemas. Em minha pesquisa, tenho investigado métodos de integração de etnografia no processo de engenharia de software por meio de sua ligação com os métodos de engenharia de requisitos (VILLER e SOMMERVILLE, 1999; VILLER e SOMMERVILLE, 2000) e documentação de padrões de interação em sistemas cooperativos (MARTIN et al., 2001; MARTIN et al., 2002; MARTIN e SOMMERVILLE, 2004).

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

1. Requisitos derivados da maneira como as pessoas realmente trabalham, e não da forma como as definições dos processos dizem que deveríam trabalhar. Por exemplo, controladores de tráfego aéreo podem desligar um sistema de alerta de conflitos que detecta aeronaves com rotas em colisão, embora os procedimentos de controle normal especifiquem que ele deve ser usado. Eles deliberadamente colocam a aeronave em caminhos conflitantes, por um curto período, para ajudar no gerenciamento do espaço aéreo. Sua estratégia de controle é projetada para assegurar que os aviões sejam afastados dessa rota conflitante antes que surjam problemas, e eles acham que o alarme de alerta distrai seu trabalho.
2. Requisitos derivados da cooperação e conhecimento das atividades de outras pessoas. Por exemplo, controladores de tráfego aéreo podem usar conhecimento do trabalho de outros controladores para prever o número de aeronaves que entrarão em seu setor de controle. Eles, então, modificam suas estratégias de controle, dependendo do volume de trabalho previsto. Portanto, um sistema ATC automatizado deve permitir aos controladores de um setor alguma visibilidade do trabalho em setores adjacentes.

A etnografia pode ser combinada com prototipação (Figura 4.7). A etnografia informa o desenvolvimento do protótipo, para que menos ciclos de refinamento do protótipo sejam necessários. Além disso, a prototipação dá foco para a etnografia, ao identificar problemas e questões que podem ser discutidos com o etnógrafo. Esse profissional deve procurar as respostas para essas perguntas durante a próxima fase do estudo do sistema (SOMMERVILLE et al., 1993).

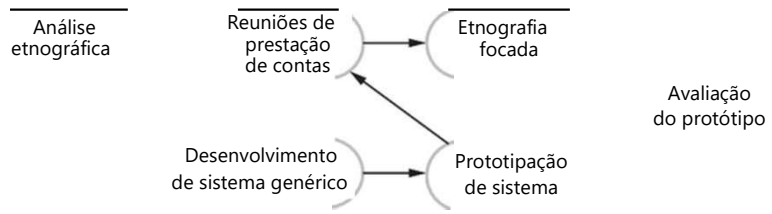
Estudos etnográficos podem revelar detalhes críticos de processo que, muitas vezes, são ignorados por outras técnicas de elicitação de requisitos. Contudo, uma vez que o foco é o usuário final, essa abordagem nem sempre é apropriada para descobrir requisitos organizacionais ou de domínio. Eles nem sempre podem identificar novos recursos que devem ser adicionados ao sistema. A etnografia, portanto, não é uma abordagem completa para elicitação e deve ser usada para complementar outras abordagens, como análise de casos de uso.

4.6 Validação de requisitos

A validação de requisitos é o processo pelo qual se verifica se os requisitos definem o sistema que o cliente realmente quer. Ela se sobrepõe à análise, uma vez que está preocupada em encontrar problemas com os requisitos. A validação de requisitos é importante porque erros em um documento de requisitos podem gerar altos custos de retrabalho quando descobertos durante o desenvolvimento ou após o sistema já estar em serviço.

O custo para consertar um problema de requisitos por meio de uma mudança no sistema é geralmente muito maior do que o custo de consertar erros de projeto ou de codificação. A razão para isso é que a ocorrência de mudança dos requisitos normalmente significa que o projeto e a implementação do sistema também devem ser alterados. Além disso, o sistema deve, posteriormente, ser retestado.

Figura 4.7 Etnografia e prototipação para análise de requisitos.



Durante o processo de validação de requisitos, diferentes tipos de verificação devem ser efetuados com os requisitos no documento de requisitos. Essas verificações incluem:

1. *Verificações de validade*. Um usuário pode pensar que é necessário um sistema para executar determinadas funções. No entanto, maior reflexão e análise mais aprofundada podem identificar funções necessárias, adicionais ou diferentes. Os sistemas têm diversos *stakeholders* com diferentes necessidades, e qualquer conjunto de requisitos é inevitavelmente um compromisso da comunidade de *stakeholders*.
2. *Verificações de consistência*. Requisitos no documento não devem entrar em conflito. Ou seja, não deve haver restrições contraditórias ou descrições diferentes da mesma função do sistema.
3. *Verificações de completude*. O documento de requisitos deve incluir requisitos que definam todas as funções e as restrições pretendidas pelo usuário do sistema.
4. *Verificações de realismo*. Usando o conhecimento das tecnologias existentes, os requisitos devem ser verificados para assegurar que realmente podem ser implementados. Essas verificações devem considerar o orçamento e o cronograma para o desenvolvimento do sistema.
5. *Verificabilidade*. Para reduzir o potencial de conflito entre o cliente e o contratante, os requisitos do sistema devem ser passíveis de verificação. Isso significa que você deve ser capaz de escrever um conjunto de testes que demonstrem que o sistema entregue atende a cada requisito especificado.

Existe uma série de técnicas de validação de requisitos que podem ser usadas individualmente ou em conjunto:

1. *Revisões de requisitos*. Os requisitos são analisados sistematicamente por uma equipe de revisores que verifica erros e inconsistências.
2. *Prototipação*. Nessa abordagem para validação, um modelo executável do sistema em questão é demonstrado para os usuários finais e clientes. Estes podem experimentar o modelo para verificar se ele atende a suas reais necessidades.
3. *Geração de casos de teste*. Os requisitos devem ser testáveis. Se os testes forem concebidos como parte do processo de validação, isso frequentemente revela problemas de requisitos. Se é difícil ou impossível projetar um teste, isso normalmente significa que os requisitos serão difíceis de serem implementados e devem ser reconsiderados. O desenvolvimento de testes a partir dos requisitos do usuário antes de qualquer código ser escrito é parte integrante do Extreme Programming.

Você não deve subestimar os problemas envolvidos na validação de requisitos. Afinal, é difícil mostrar que um conjunto de requisitos atende de fato às necessidades de um usuário; os usuários precisam imaginar o sistema em operação e como esse sistema se encaixaria em seu trabalho. Até para profissionais qualificados de informática é difícil fazer esse tipo de análise abstrata, e é mais difícil ainda para os usuários do sistema. Como resultado, durante o processo de validação dos requisitos você raramente encontrará todos os problemas de requisitos. Após o ajuste do documento de requisitos, é inevitável a necessidade de mudanças nos requisitos para corrigir omissões e equívocos.

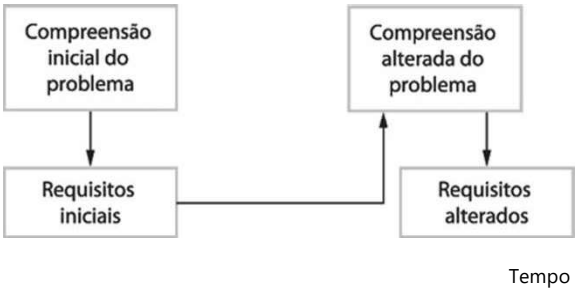
^ 4.7 Gerenciamento de requisitos

Os requisitos para sistemas de software de grande porte estão sempre mudando. Uma razão para isso é que esses sistemas geralmente são desenvolvidos para enfrentar os problemas'maus'— problemas que não podem ser completamente definidos. Porque o problema não pode ser totalmente definido, os requisitos de software são obrigados a ser incompletos. Durante o processo de software, o entendimento dos *stakeholders* a respeito do problema está em constante mutação (Figura 4.8). Logo, os requisitos de sistema devem evoluir para refletir essas novas percepções do problema.

Uma vez que um sistema tenha sido instalado e seja usado regularmente, inevitavelmente surgirão novos requisitos. É difícil para os usuários e clientes do sistema anteciparem os efeitos que o novo sistema terá sobre seus processos de negócio e sobre a forma que o trabalho é realizado. Quando os usuários finais tiverem a experiência de um sistema, descobrirão novas necessidades e prioridades. Existem várias razões pelas quais a mudança é inevitável:

1. Após a instalação, o ambiente técnico e de negócios do sistema sempre muda. Um novo hardware pode ser introduzido, pode ser necessário fazer a interface do sistema com outros sistemas, as prioridades do negócio podem mudar (com consequentes alterações necessárias no apoio do sistema), podem ser introduzidas novas legislações e regulamentos aos quais o sistema deve, necessariamente, respeitar etc.

Figura 4.8 Evolução dos requisitos.



2. As pessoas que pagam por um sistema e os usuários desse sistema raramente são os mesmos. Clientes do sistema impõem requisitos devido a restrições orçamentárias e organizacionais, os quais podem entrar em conflito com os requisitos do usuário final, e, após a entrega, novos recursos podem ser adicionados, para dar suporte ao usuário, a fim de que o sistema cumpra suas metas.
3. Geralmente, sistemas de grande porte têm uma comunidade de diversos usuários, com diferentes requisitos e prioridades, que podem ser conflitantes ou contraditórios. Os requisitos do sistema final são, certamente, um compromisso entre esses usuários, e, com a experiência, frequentemente se descobre que o balanço de apoio prestado aos diferentes usuários precisa ser mudado.

O gerenciamento de requisitos é o processo de compreensão e controle das mudanças nos requisitos do sistema. Você precisa se manter a par das necessidades individuais e manter as ligações entre as necessidades dependentes para conseguir avaliar o impacto das mudanças nos requisitos. Você precisa estabelecer um processo formal para fazer propostas de mudanças e a ligação destas às exigências do sistema. O processo formal de gerenciamento de requisitos deve começar assim que uma versão preliminar do documento de requisitos estiver disponível. No entanto, você deve começar a planejar como gerenciar mudanças de requisitos durante o processo de elicitação de requisitos.

4<7*1 Planejamento de gerenciamento de requisitos

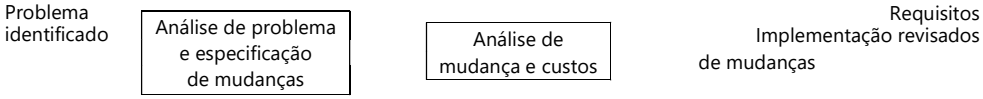
O planejamento é o primeiro estágio essencial no processo de gerenciamento de requisitos, e determina o nível de detalhamento requerido no gerenciamento de requisitos. Durante o estágio de gerenciamento de requisitos, você deve decidir sobre:

1. *Identificação de requisitos*. Cada requisito deve ser identificado unicamente para poder ser comparado com outros requisitos e usado em avaliações de rastreabilidade.
2. *Processo de gerenciamento de mudanças*. Esse é o conjunto de atividades que avaliam o impacto e o custo das mudanças. Na próxima seção, vou discutir detalhadamente esse processo.
3. *Políticas de rastreabilidade*. Definem os relacionamentos entre cada requisito e entre os requisitos e o projeto de sistema que deve ser registrado. A política de rastreabilidade também deve definir como esses registros devem ser mantidos.
4. *Ferramenta de apoio*. Gerenciamento de requisitos envolve o processamento de grandes quantidades de informações sobre os requisitos. Ferramentas que podem ser usadas variam desde sistemas especializados em gerenciamento de requisitos até planilhas e sistemas de banco de dados simples.

O gerenciamento de requisitos precisa de apoio automatizado, e as ferramentas de software para esse gerenciamento devem ser escolhidas durante a fase de planejamento. Você precisa de ferramentas de apoio para:

1. *Armazenamento de requisitos*. Os requisitos devem ser mantidos em um repositório de dados gerenciado e seguro, acessível a todos os envolvidos no processo de engenharia de requisitos.
2. *Gerenciamento de mudanças*. O processo de gerenciamento de mudanças (Figura 4.9) é simplificado quando as ferramentas ativas de apoio estão disponíveis.
3. *Gerenciamento de rastreabilidade*. Como discutido anteriormente, as ferramentas de apoio para rastreabilidade permitem descobrir requisitos relacionados. Algumas ferramentas estão disponíveis, as quais usam técnicas de processamento de linguagem natural para ajudar a descobrir as possíveis relações entre os requisitos.

Figura 4.9 Gerenciamento de mudança de requisitos.



Para sistemas de pequeno porte, podem não ser necessárias ferramentas especializadas no gerenciamento de requisitos. O processo de gerenciamento de requisitos pode ser apoiado por recursos disponíveis nos processadores de texto, planilhas e bancos de dados do PC. No entanto, para sistemas maiores, ferramentas de apoio mais especializadas são necessárias. Incluí links para informações sobre as ferramentas de gerenciamento de requisitos nas páginas do livro no site.

4.7.2 Gerenciamento de mudança de requisitos

Após a aprovação do documento de requisitos, o gerenciamento de mudança de requisitos (Figura 4.9) deve ser aplicado a todas as mudanças propostas aos requisitos de um sistema. O gerenciamento de mudanças é essencial, pois é necessário decidir se os benefícios da implementação de novos requisitos justificam os custos de implementação. A vantagem de se usar um processo formal de gerenciamento de mudanças é que todas as propostas de mudanças são tratadas de forma consistente, e as alterações nos documentos de requisitos são feitas de forma controlada.

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

1. *Análise de problema e especificação de mudanças.* O processo começa com um problema de requisitos identificado ou, às vezes, com uma proposta específica de mudança. Durante esse estágio, analisa-se o problema ou a proposta de mudança a fim de se verificar sua validade. Essa análise é transmitida a quem solicitou a mudança, que pode responder com uma proposta mais específica de mudança de requisitos ou retirar a solicitação.
2. *Análise de mudanças e custos.* O efeito da mudança proposta é avaliado por meio de informações de rastreabilidade e conhecimentos gerais dos requisitos do sistema. O custo de fazer a mudança é estimado em termos de modificações no documento de requisitos e, se apropriado, no projeto e implementação do sistema. Uma vez que essa análise é concluída, decide-se prosseguir ou não com a mudança de requisitos.
3. *Implementação de mudanças.* O documento de requisitos e, quando necessário, o projeto e implementação do sistema são modificados. Você deve organizar o documento de requisitos para poder fazer alterações sem ampla reformulação ou reorganização.Tal como acontece com os programas, a mutabilidade nos documentos é obtida minimizando-se as referências externas e tornando as seções do documento o mais modular possível. Assim, as seções individuais podem ser alteradas e substituídas sem afetar outras partes do documento.

Se um novo requisito precisa ser implementado com urgência, há sempre a tentação de mudar o sistema e, em seguida, retrospectivamente modificar o documento de requisitos. Esse procedimento deve ser evitado, pois quase inevitavelmente faz com que a especificação de requisitos e a implementação do sistema fiquem defasadas. Uma vez que mudanças no sistema foram feitas, é fácil esquecer de incluir essas alterações no documento de requisitos ou acrescentar informações inconsistentes com a implementação no documento de requisitos.

Processos ágeis de desenvolvimento, como Extreme Programming, foram concebidos para lidar com requisitos mutáveis durante o processo de desenvolvimento. Nesses processos, quando um usuário propõe uma mudança nos requisitos, a mudança não passa por um processo formal de gerenciamento de mudanças. Pelo contrário, o usuário tem de priorizar essa mudança e, em caso de alta prioridade, decidir quais recursos do sistema planejados para a próxima iteração devem ser abandonados.

^ PONTOS IMPORTANTES

- Os requisitos para um sistema de software estabelecem o que o sistema deve fazer e define as restrições sobre seu funcionamento e implementação.
- Os requisitos funcionais são declarações dos serviços que o sistema deve fornecer ou descrições de como alguns processamentos devem ser efetuados.
- Muitas vezes, os requisitos não funcionais restringem o sistema que está sendo desenvolvido e o processo de

desenvolvimento que está sendo usado. Estes podem ser os requisitos de produto, requisitos organizacionais ou requisitos externos. Eles costumam se relacionar com as propriedades emergentes do sistema e, portanto, aplicam-se ao sistema como um todo.

- O documento de requisitos de software é uma declaração acordada dos requisitos do sistema. Deve ser organizado para que ambos — os clientes do sistema e os desenvolvedores de software — possam usá-lo.
- O processo de engenharia de requisitos inclui um estudo da viabilidade, elicitação e análise de requisitos, especificação de requisitos, validação e gerenciamento de requisitos.
- Elicitação e análise de requisitos é um processo iterativo que pode ser representado como uma espiral de atividades — descoberta de requisitos, classificação e organização de requisitos, negociação de requisitos e documentação de requisitos.
- A validação de requisitos é o processo de verificação da validade, consistência, completude, realismo e verificabilidade dos requisitos.
- Mudanças organizacionais, mudanças nos negócios e mudanças técnicas inevitavelmente geram mudanças nos requisitos para um sistema de software. O gerenciamento de requisitos é o processo de gerenciamento e controle dessas mudanças.

&LEITURA COMPLEMENTAR ^

Software Requirements, 2nd edition. Esse livro, projetado para escritores e usuários de requisitos, discute boas práticas de engenharia de requisitos. (WEIGERS, K. M. *Software Requirements*. 2. ed. Microsoft Press., 2003.)

Integrated requirements engineering: A tutorial. Esse é um artigo tutorial que escrevi, no qual discuto as atividades da engenharia de requisitos e como elas podem ser adaptadas para as práticas modernas da engenharia de software. (SOMMERVILLE, I. Integrated requirements engineering: A tutorial. *IEEE Software,* v. 22, n. 1, jan.-fev. 2005.) Disponível em: <http://dx.doi.Org/10.1109/MS.2005.13>.

Mastering theRequirements Process, 2ndedition. Um livro bem escrito e fácil de ler, que se baseia em um método específico (VOLERE), mas que também inclui bons conselhos gerais sobre engenharia de requisitos. (ROBERTSON, S.; ROBERTSON, J. *Mastering the Requirements Process. 2.* ed. Addison-Wesley, 2006.)

Research Directions in Requirements Engineering. Esse é um bom levantamento da pesquisa de engenharia de requisitos, que destaca os desafios das futuras pesquisas da área para resolver questões como escala e agilidade. (CHENG, B. H. C.; ATLEE, J. M. Research Directions in Requirements Engineering. *Proc. Confon Future of Software Engineering,* IEEE Computer Society, 2007.) Disponível em: <http://dx.doi.Org/10.1109/FOSE.2007.17>.

MEEXERCÍCIOS

4.1 Identifique e descreva brevemente os quatro tipos de requisitos que podem ser definidos para um sistema computacional.

4.2 Descubra ambiguidades ou omissões nas seguintes declarações de requisitos para parte de um sistema de emissão de bilhetes:

Um sistema automatizado para emitir bilhetes vende bilhetes de trem. Os usuários selecionam seu destino e inserem um cartão de crédito e um número de identificação pessoal. O bilhete é emitido, e sua conta de cartão de crédito, cobrada. Quando o usuário pressiona o botão de início, é ativado um *dispiay* de menu de destinos possíveis, junto com uma mensagem ao usuário para selecionar um destino. Uma vez que o destino tenha sido selecionado, os usuários são convidados a inserir seu cartão de crédito. Sua validade é verificada e, em seguida, é solicitada ao usuário a entrada de um identificador pessoal. Quando a operação de crédito é validada, o bilhete é emitido.

4.3 Reescreva a descrição anterior usando a abordagem estruturada descrita neste capítulo. Resolva, de um modo apropriado, as ambiguidades identificadas.

4.4 Escreva um conjunto de requisitos não funcionais para o sistema de emissão de bilhetes, definindo sua confiabilidade e tempo de resposta esperados.

4.5 Usando a técnica sugerida neste capítulo, em que as descrições em linguagem natural são apresentadas em formato-padrão, escreva requisitos do usuário plausíveis para as seguintes funções:

- Um sistema de bomba de gasolina autônoma, que inclui um leitor de cartão de crédito. 0 cliente passa o cartão pelo leitor e, em seguida, especifica a quantidade de combustível requerida. O combustível é liberado, e a conta do cliente, debitada.
- A função de distribuidor de dinheiro em um caixa eletrônico de banco (ATM).
- Os recursos de verificação e correção ortográfica em um editor de texto.

4.6 Sugira como um engenheiro responsável pela elaboração de um sistema de especificação de requisitos pode manter o acompanhamento dos relacionamentos entre requisitos funcionais e não funcionais.

4.7 Usando seu conhecimento de como um caixa eletrônico (ATM) funciona, desenvolva um conjunto de casos de uso que podería servir de base para o entendimento dos requisitos para um sistema de ATM.

4.8 Quem deve ser envolvido em uma revisão de requisitos? Desenhe um modelo de processo mostrando como uma revisão de requisitos pode ser organizada.

4.9 Quando mudanças emergenciais precisam ser feitas em sistemas, o software do sistema pode precisar ser modificado antes de serem aprovadas as mudanças nos requisitos. Sugira um modelo de um processo para fazer essas modificações de modo a garantir que o documento de requisitos e implementação do sistema não se tornem inconsistentes.

4.10 Você está trabalhando com um usuário de software que contratou seu empregador anterior; juntos, buscam desenvolver um sistema para ele. Você descobre que a interpretação dos requisitos por sua empresa atual é diferente da interpretação de seu empregador anterior. Discuta o que você deve fazer em tal situação. Você sabe que os custos para seu atual empregador aumentarão se as ambiguidades não forem resolvidas. No entanto, você também tem a responsabilidade da confidencialidade com seu empregador anterior.

REFERÊNCIAS

BECK, K. Embracing Change with Extreme Programming. *IEEE Computer*, v. 32, n. 10,1999, p. 70-78.

CRABTREE, A. *Designing CollaborativeSystems:* A Practical Guide to Ethnography. Londres: Springer-Verlag, 2003.

DAVIS, A. M. *Software Requirements: Objects, Functions and States.* Englewood Cliffs, NJ: Prentice Hall, 1993.

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: Addison-Wesley, 1993.

KOTONYA, G.; SOMMERVILLE, I. *Requirements Engineering:* Processes and Techniques. Chichester, Reino Unido: John Wileyand Sons, 1998.

LARMAN, C. *Applying UML and Patterns: An Introduction to Object-OrientedAnalysis and Design and the Unified Process.* Englewood Cliff, NJ: Prentice Hall, 2002.

MARTIN, D.; RODDEN,T; ROUNCEFIELD, M.; SOMMERVILLE, I.; VILLER, S. Finding Patterns in the Fieldwork. *Proc. ECSCW'01.* Bonn: Kluwer, 2001, p. 39-58.

MARTIN, D.; ROUNCEFIELD, M.; SOMMERVILLE, I. Applying patterns of interaction to work (re)design: E-government and planning. *Proc ACM CHI'2002,* ACM Press, 2002, p. 235-242.

MARTIN, D.; SOMMERVILLE, I. Patterns of interaction: Linking ethnomethodology and design. *ACM Trans. on Computer-Human Interaction,* v. 11, n. 1,2004, p. 59-89.

ROBERTSON, S.; ROBERTSON, J. *Mastering the Requirements Process.* Harlow, Reino Unido: Addison-Wesley, 1999.

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

STEVENS, R; POOLEY, R. *Using UML: Software Engineering with Objects and Components. 2.* ed. Harlow, Reino Unido: Addison-Wesley, 2006.

SUCHMAN, L. *Plans andSituatedActions.* Cambridge: Cambridge University Press, 1987.

VILLER, S.; SOMMERVILLE, I. Coherence: An Approach to Representing Ethnographic Analyses in Systems Design. *Human-Computer Interaction,* v. 14, n. 1,2,1999, p. 9-41.

_____ . Ethnographically informed analysis for software engineers. *Int. J. of Human-Computer Studies,* v. 53, n. 1, 2000, p. 169-196.