

5

ENGENHARIA DE REQUISITOS

CONCEITOS-CHAVE

casos de uso	137
colaboração	132
concepção	127
disponibilização da função de qualidade ..	136
elaboração	128
engenharia de requisitos	127
especificação	129
gestão dos requisitos	130
interessados	131
levantamento	128
levantamento dos requisitos	133
modelo de análise ..	142
negociação	128
padrões de análise ..	145
pontos de vista	131
produtos resultantes	137
validação	129
validação dos requisitos	146

Entender os requisitos de um problema está entre as tarefas mais difíceis enfrentadas por um engenheiro de software. Ao pensar nisso pela primeira vez, a engenharia de requisitos não parece assim tão difícil. Afinal de contas, o cliente não sabe o que é necessário? Os usuários finais não deveriam ter um bom entendimento das características e funções que trarão benefícios? Surpreendentemente, em muitos casos a resposta a essas perguntas é “não”. E mesmo se os clientes e usuários finais fossem explícitos quanto às suas necessidades, essas mudariam ao longo do projeto.

No prefácio de um livro de Ralph Young [You01] sobre práticas de necessidades eficazes, escreveu:

É o seu pior pesadelo. Um cliente entra no seu escritório, se senta, te olha diretamente nos olhos e diz: “Eu sei que você imagina que entendeu aquilo que eu lhe disse, mas o que você não compreende é que aquilo que eu lhe disse não era exatamente aquilo que eu quis dizer”. Invariavelmente, isso acontece no final do projeto, após compromissos de prazos de entrega terem sido estabelecidos, reputações estarem na mira e muito dinheiro estar em jogo.

Todos que já trabalharam na área de software e sistemas há alguns anos viveram esse pesadelo e, mesmo assim, poucos aprenderam a lidar-se dele. Passamos por muitas dificuldades ao tentar extrair os requisitos de nossos clientes. Temos dificuldades para entender as informações obtidas. Normalmente registramos os requisitos de uma forma desorganizada e investimos muito pouco tempo verificando aquilo que registramos. Deixamos que as mudanças nos controlem, em vez de estabelecermos mecanismos para controlar as mudanças. Em suma, falhamos ao estabelecer uma base sólida para o sistema ou software. Cada um desses problemas é desafiador. Quando combinados, o panorama é assustador até mesmo para os gerentes e profissionais mais experientes. Mas soluções de fato existem.

PANORAMA

O que é? Antes de iniciar qualquer trabalho técnico, é uma boa ideia aplicar um conjunto de tarefas de engenharia de requisitos. Estas levam a um entendimento de qual será o impacto do software sobre o negócio, o que o cliente quer e como os usuários finais irão interagir com o software.

Quem realiza? Engenheiros de software (algumas vezes conhecidos no mundo da TI como engenheiros de sistemas ou “analistas”) e outros interessados no projeto (gerentes, clientes, usuários finais), todos participam da engenharia de requisitos.

Por que é importante? Projetar e construir um programa de computador elegante que resolva o problema errado não atende às necessidades de ninguém. É por isso que é importante entender o que o cliente quer antes de começar a projetar e construir um sistema baseado em computador.

Quais são as etapas envolvidas? A engenharia de requisitos começa com a concepção — uma tarefa que define o escopo e a natureza do problema a ser resolvido. Ela prossegue para o levantamento — uma tarefa que ajuda os interessados a definir o que é necessário e, então, para a elabo-

ração — onde os requisitos básicos são refinados e modificados. À medida que os interessados definem o problema, ocorre a negociação — quais são as prioridades, o que é essencial, quando é necessário? Por fim, o problema é especificado de algum modo e, então, revisado ou validado para garantir que seu entendimento e o entendimento dos interessados sobre o problema coincidam.

Qual é o artefato? O objetivo da engenharia de requisitos é fornecer a todas as partes um entendimento escrito do problema. Isso pode ser alcançado por meio de uma série de artefatos: cenários de uso, listas de funções e características, modelos de análise ou uma especificação.

Como garantir que o trabalho foi realizado corretamente? Os artefatos da engenharia de requisitos são revisados com os interessados para garantir que aquilo que você entendeu é realmente aquilo que eles queriam dizer. Um alerta: mesmo depois de todas as partes terem entrado em acordo, as coisas vão mudar e continuarão mudando ao longo de todo o projeto.

É razoável argumentar que as técnicas que discutiremos neste capítulo não são uma verdadeira “solução” para os desafios que acabamos de citar. Mas elas fornecem efetivamente uma abordagem consistente para lidar com tais desafios.

5.1 ENGENHARIA DE REQUISITOS

“A parte mais difícil ao construir um sistema de software é decidir o que construir. Nenhuma parte do trabalho afeta tanto o sistema resultante se for feita a coisa errada. Nenhuma outra parte é mais difícil de consertar depois.”

Fred Brooks

PONTO-CHAVE

A engenharia de requisitos estabelece uma base sólida para o projeto e para a construção. Sem ela, o software resultante tem grande probabilidade de não atender às necessidades do cliente.



Espreze ter de realizar um pouco de projeto durante o trabalho de levantamento de requisitos e um pouco de trabalho de levantamento de requisitos durante o projeto.

Projetar e construir software é desafiador, criativo e pura diversão. Na realidade, construir software é tão cativante que muitos desenvolvedores desejam iniciar logo, antes de terem um claro entendimento daquilo que é necessário. Eles argumentam que as coisas ficarão mais claras à medida que forem construindo o software, que os interessados no projeto serão capazes de entender a necessidade apenas depois de examinar as primeiras iterações do software, que as coisas mudam tão rápido que qualquer tentativa de entender os requisitos de forma detalhada será uma perda de tempo, que o primordial é produzir um programa que funcione e que todo o resto é secundário. O que torna esses argumentos tentadores é que contêm elementos de verdade.¹ Porém, cada um apresenta pontos fracos e pode levar um projeto ao fracasso.

O amplo espectro de tarefas e técnicas que levam a um entendimento dos requisitos é denominado *engenharia de requisitos*. Na perspectiva do processo de software, a engenharia de requisitos é uma ação de engenharia de software importante que se inicia durante a atividade de comunicação e continua na de modelagem. Ela deve ser adaptada às necessidades do processo, do projeto, do produto e das pessoas que estão realizando o trabalho.

A engenharia de requisitos constrói uma ponte para o projeto e para a construção. Mas onde começa essa ponte? Alguém pode argumentar que ela começa na base dos interessados no projeto (por exemplo, gerentes, clientes, usuários finais), em que é definida a necessidade do negócio, são descritos cenários de usuários, delineados funções e recursos e identificadas restrições de projeto. Outros poderiam sugerir que ela se inicia com uma definição mais abrangente do sistema, em que o software é apenas um componente do domínio de sistema mais abrangente. Porém, independentemente do ponto de partida, a jornada através da ponte nos leva bem à frente no projeto, permitindo que examinemos o contexto do trabalho de software a ser realizado; as necessidades específicas que o projeto e a construção devem atender; as prioridades que orientam a ordem na qual o trabalho deve ser completado e as informações, funções e comportamentos que terão um impacto profundo no projeto resultante.


A engenharia de requisitos fornece o mecanismo apropriado para entender aquilo que o cliente deseja, analisando as necessidades, avaliando a viabilidade, negociando uma solução razoável, especificando a solução sem ambiguidades, validando a especificação e gerenciando as necessidades à medida que são transformadas em um sistema operacional [Tha97]. Ela abrange sete tarefas distintas: concepção, levantamento, elaboração, negociação, especificação, validação e gestão. É importante notar que algumas delas ocorrem em paralelo e todas são adaptadas às necessidades do projeto.

Concepção. Como um projeto de software é iniciado? Existe algum evento único que se torna o catalisador para um novo produto ou sistema de computador ou a necessidade evolui ao longo do tempo? Não há nenhuma resposta definitiva para tais perguntas. Em alguns casos, uma conversa informal é tudo o que é preciso para precipitar um trabalho de engenharia de software. Porém, em geral, a maioria dos projetos começa quando é identificada a necessidade do negócio ou é descoberto um novo serviço ou mercado potencial. Interessados da comunidade de negócios (por exemplo, gerentes comerciais, pessoal de marketing, gerentes de produto) definem um plano de negócio para a ideia, tentam identificar o tamanho do mercado, fazem uma análise de

¹ Isto é particularmente verdadeiro para projetos pequenos (menos de um mês) e trabalhos de software menores e relativamente simples. À medida que o software cresce em tamanho e complexidade, tais argumentos caem por terra.

"As sementes das principais catástrofes de software são normalmente semeadas nos três primeiros meses do projeto de software."

Caper Jones

 **Por que é difícil conseguir um claro entendimento daquilo que o cliente quer?**

 **AVISO**

A elaboração é uma coisa boa, porém é preciso saber quando parar. O segredo é descrever o problema de maneira que estabeleça uma base sólida para o projeto. Caso passe desse ponto, você estará realizando um projeto.

 **AVISO**

Em uma negociação efetiva não existem ganhadores, nem perdedores. Ambas as partes ganham, pois é solidificado um "trato" que ambas as partes aceitam.

viabilidade aproximada e identificam uma descrição operacional do escopo do projeto. Todas as informações estão sujeitas a mudanças, porém é suficiente para suscitar discussões com a organização de engenharia de software.²

Na concepção³ do projeto, estabelecemos um entendimento básico do problema, as pessoas que querem uma solução, a natureza da solução desejada e a eficácia da comunicação e colaboração preliminares entre os demais interessados e a equipe de software.

Levantamento. Certamente parece bastante simples — pergunte ao cliente, aos usuários e aos demais interessados quais são os objetivos para o sistema ou produto, o que deve ser alcançado, como o sistema ou produto atende às necessidades da empresa e, por fim, como o sistema ou produto deve ser utilizado no dia a dia. Mas isso não é simples — na verdade, é muito difícil.

Christel e Kang [Cri92] identificaram uma série de problemas que são encontrados durante o levantamento:

- **Problemas de escopo.** Os limites do sistema são definidos de forma precária ou os clientes/usuários especificam detalhes técnicos desnecessários que podem confundir, em vez de esclarecer, os objetivos globais do sistema.
- **Problemas de entendimento.** Os clientes/usuários não estão completamente certos do que é preciso, têm um entendimento inadequado das capacidades e limitações de seus ambientes computacionais, não possuem um entendimento completo do domínio do problema, têm problemas para transmitir suas necessidades ao engenheiro de sistemas, omitem informações que acreditam ser "óbvias", especificam requisitos que conflitam com as necessidades de outros clientes/usuários ou especificam requisitos que são ambíguos ou impossíveis de ser testados.
- **Problemas de volatilidade.** Os requisitos mudam com o tempo. Para ajudar a superar esses problemas, devemos abordar o levantamento de requisitos de forma organizada.

Elaboração. As informações obtidas do cliente durante as fases de concepção e levantamento são expandidas e refinadas durante a elaboração. Essa tarefa concentra-se no desenvolvimento de um modelo de requisitos refinado (Capítulos 6 e 7) que identifique os diversos aspectos da função, do comportamento e das informações do software.

A elaboração é guiada pela criação e refinamento de cenários de usuários que descrevem como o usuário final (e outros atores) irão interagir com o sistema. Cada cenário de usuário é analisado sintaticamente para extrair classes de análise — entidades do domínio de negócio visíveis para o usuário final. Os atributos de cada classe de análise são definidos, e os serviços⁴ exigidos por cada classe são identificados. As relações e a colaboração entre as classes são identificadas e uma variedade de diagramas suplementares é produzida.

Negociação. Não é incomum clientes e usuários pedirem mais do que pode ser alcançado, dados os recursos limitados do negócio. Também é relativamente comum diferentes clientes ou usuários proporem necessidades conflitantes, argumentando que sua versão é "essencial para nossas necessidades especiais".

É preciso conciliar esses conflitos por meio de um processo de negociação. Devemos solicitar aos clientes, usuários e outros interessados para que ordenem seus requisitos e discutam em termos de prioridade. Usando uma abordagem iterativa que priorize os requisitos, avalie seus

² Se um sistema baseado em computador deve ser desenvolvido, as discussões começam no contexto de um processo de engenharia de sistemas. Para uma discussão detalhada da engenharia de sistemas, visite o site que acompanha este livro.

³ Lembre-se de que o Processo Unificado (Capítulo 2) define uma "fase de concepção" mais ampla que engloba as tarefas de concepção, levantamento e elaboração discutidas neste capítulo.

⁴ Um serviço manipula os dados encapsulados pela classe. Os termos *operação* e *método* também são usados. Caso não esteja familiarizado com os conceitos da orientação a objetos, é apresentada uma introdução básica no Apêndice 2.

custos e riscos, bem como trate dos conflitos internos, requisitos são eliminados, combinados e/ou modificados, de modo que cada parte atinja certo nível de satisfação.

Especificação. No contexto de sistemas (e software) baseados em computadores, o termo *especificação* assume diferentes significados para pessoas diferentes. Especificação pode ser um documento por escrito, um conjunto de modelos gráficos, um modelo matemático formal, um conjunto de cenários de uso, um protótipo ou qualquer combinação dos fatores citados.

Alguns sugerem que “modelo-padrão” [Som97] deve ser desenvolvido e utilizado para a especificação, argumentando que isso leva a requisitos que são apresentados de forma consistente e, portanto, mais compreensível. Entretanto, algumas vezes é necessário permanecer flexível quando uma especificação deve ser desenvolvida. Para sistemas grandes, um documento escrito, combinando descrições em linguagem natural e modelos gráficos, pode ser a melhor abordagem. Entretanto, talvez sejam necessários apenas cenários de uso para produtos ou sistemas menores que residem em ambientes técnicos bem entendidos.

PONTO-CHAVE

A formalidade e o formato de uma especificação variam com o tamanho e a complexidade do software a ser construído.



Modelo de especificação de requisitos de software

Uma especificação de requisitos de software (*software requirements specification*, SRS) é um documento criado quando uma descrição detalhada de todos os aspectos do software a ser construído deve ser especificada antes de o projeto começar. É importante notar que uma SRS formal nem sempre é por escrito. Na realidade, há várias ocasiões em que o esforço gasto em uma SRS talvez fosse mais bem aproveitado em outras atividades de engenharia de software. Entretanto, quando um software for desenvolvido por terceiros, quando uma falta de especificação criar graves problemas de negócio, ou quando um sistema for extremamente complexo ou crítico para o negócio, será justificável uma SRS.

Karl Wiegers [Wie03] da Process Impact Inc. desenvolveu uma planilha bastante útil (disponível em www.processimpact.com/process_assets/srs_template.doc), que pode servir como diretriz para aqueles que precisam criar uma SRS completa. Uma descrição geral por tópicos é apresentada a seguir:

Sumário

Histórico de revisão

1. Introdução

- 1.1 Propósito
- 1.2 Convenções do documento
- 1.3 Público-alvo e sugestões de leitura
- 1.4 Escopo do projeto
- 1.5 Referências

2. Descrição geral

- 2.1 Perspectiva do produto
- 2.2 Características do Produto

2.3 Classes de usuários e características

- 2.4 Ambiente operacional
- 2.5 Restrições de projeto e implementação
- 2.6 Documentação para usuários
- 2.7 Hipóteses e dependências

3. Características do sistema

- 3.1 Características do sistema 1
- 3.2 Características do sistema 2 (e assim por diante)

4. Requisitos de interfaces externas

- 4.1 Interfaces do usuário
- 4.2 Interfaces de hardware
- 4.3 Interfaces de software
- 4.4 Interfaces de comunicação

5. Outros requisitos não funcionais

- 5.1 Necessidades de desempenho
- 5.2 Necessidades de proteção
- 5.3 Necessidades de segurança
- 5.4 Atributos de qualidade de software

6. Outros requisitos

Apêndice A: Glossário

Apêndice B: Modelos de análise

Apêndice C: Lista de problemas

Uma descrição detalhada de cada tópico SRS pode ser obtida fazendo-se o download da planilha SRS na URL citada anteriormente neste quadro.

Validação. Os artefatos produzidos como consequência da engenharia de requisitos são avaliados quanto à qualidade durante a etapa de validação. A validação de requisitos examina a especificação⁵ para garantir que todos os requisitos de software tenham sido declarados de for-

⁵ Lembre-se de que a natureza da especificação irá variar em cada projeto. Em alguns casos, a “especificação” é um conjunto de cenários de usuários e pouco mais do que isso. Em outros, a especificação pode ser um documento contendo cenários, modelos e descrições por escrito.

INFORMAÇÕES



Uma preocupação fundamental durante a validação de requisitos é a consistência. Use o modelo de análise para garantir que os requisitos foram declarados de forma consistente.

ma não ambígua; que as inconsistências, omissões e erros tenham sido detectados e corrigidos e que os artefatos estejam de acordo com os padrões estabelecidos para o processo, projeto e produto.

O principal mecanismo de validação de requisitos é a revisão técnica (Capítulo 15). A equipe de revisão que valida os requisitos é formada por engenheiros de software, clientes, usuários e outros interessados que examinam a especificação em busca de erros no conteúdo ou na interpretação, áreas em que talvez sejam necessários esclarecimentos, de informações faltantes, de inconsistências (um problema grave quando são criados produtos ou sistemas grandes), de requisitos conflitantes ou de requisitos irreais (inatingíveis).

INFORMAÇÕES



Lista de controle para validação de requisitos

Muitas vezes é útil examinar cada requisito em relação a um conjunto de perguntas contidas em uma lista de controle. A seguir, um pequeno subconjunto daquelas que poderiam ser perguntadas:

- Os requisitos estão declarados de forma clara? Eles podem ser mal-interpretados?
- A fonte (por exemplo, uma pessoa, uma regulamentação, um documento) do requisito foi identificada? A declaração final do requisito foi examinada pela fonte original ou com ela?
- O requisito está limitado em termos quantitativos?
- Que outros requisitos se relacionam a este requisito? Eles estão claramente indicados por meio de uma matriz de referência cruzada ou algum outro mecanismo?
- O requisito viola quaisquer restrições do domínio do sistema?
- O requisito pode ser testado? Em caso positivo, podemos especificar testes (algumas vezes denominados critérios de validação) para testar o requisito?
- O requisito pode ser associado a qualquer modelo de sistema que tenha sido criado?
- O requisito pode ser associado aos objetivos globais do sistema/produto?
- A especificação é estruturada de forma que leve ao fácil entendimento, fácil referência e fácil tradução em artefatos mais técnicos?
- Criou-se um índice para a especificação?
- Os requisitos associados ao desempenho, ao comportamento e às características operacionais foram declarados de maneira clara? Quais requisitos parecem estar implícitos?

Gestão de requisitos. Os requisitos para sistemas baseados em computadores mudam, e o desejo de mudar os requisitos persiste ao longo da vida de um sistema. Gestão de requisitos é um conjunto de atividades que ajuda a equipe de projeto a identificar, controlar e acompanhar as necessidades e suas mudanças a qualquer momento enquanto o projeto prossegue.⁶ Muitas

FERRAMENTAS DO SOFTWARE



Engenharia de requisitos

Objetivo: As ferramentas de engenharia de requisitos auxiliam no levantamento, na modelagem, na gestão, bem como na validação de requisitos.

Mecânica: A mecânica das ferramentas varia. Em geral, as ferramentas de engenharia de requisitos constroem uma grande variedade de modelos gráficos (por exemplo, UML) que representam os aspectos informativos, funcionais e comportamentais de um sistema. Esses modelos formam a base para todas as demais atividades no processo de software.

Ferramentas representativas:⁷

Uma lista relativamente abrangente (e atualizada) de ferramentas de engenharia de requisitos pode ser encontrada no site Volere Requirements em www.volere.co.uk/tools.htm. As ferramentas de modelagem de requisitos são discutidas nos

Capítulos 6 e 7. As ferramentas citadas a seguir se concentram na gestão de requisitos.

EasyRM, desenvolvida pela Cybernetic Intelligence GmbH (www.easy-rm.com), constrói um dicionário/glossário específico de projetos contendo atributos e descrições detalhadas dos requisitos.

Rational RequisitePro, desenvolvida pela Rational Software (www-306.ibm.com/software/awdtools/reqpro/), permite aos usuários construir um banco de dados de requisitos, representar as relações entre requisitos e organizar, priorizar e rastrear requisitos.

Muitas outras ferramentas de gerenciamento de necessidades podem ser encontradas no site da Volere citado anteriormente no seguinte endereço: www.jiludwig.com/Requirements_Management_Tools.html.

⁶ O gerenciamento formal de requisitos é iniciado apenas para grandes projetos com centenas de necessidades identificáveis. Para projetos pequenos, essa ação da engenharia de requisitos é consideravelmente menos formal.

⁷ As ferramentas aqui apresentadas não significam um endosso, mas sim, uma amostra de ferramentas desta categoria. Na maioria dos casos, os nomes das ferramentas são marcas registradas por seus respectivos desenvolvedores.

dessas atividades são idênticas às técnicas de gerenciamento de configurações de software (*software configuration management*, SCM) discutidas no Capítulo 22.

5.2 INÍCIO DO PROCESSO DE ENGENHARIA DE REQUISITOS

PONTO-CHAVE

Interessado é qualquer um que tenha interesse direto ou benefício do sistema a ser desenvolvido.

Em um ambiente ideal, os interessados e os engenheiros de software trabalham juntos na mesma equipe.⁸ Em tais casos, a engenharia de requisitos é apenas uma questão de conduzir conversações proveitosas com colegas que sejam membros bem conhecidos da equipe. Porém, a realidade muitas vezes é bastante diferente.

Cliente(s) ou usuários finais podem estar situados em uma cidade ou país diferente, podem ter apenas uma vaga ideia daquilo que é necessário, podem ter opiniões conflitantes sobre o sistema a ser construído, podem ter conhecimento técnico limitado ou quem sabe pouco tempo para interagir com o engenheiro que está fazendo o levantamento dos requisitos. Nenhuma das situações anteriores é desejável, contudo, todas são relativamente comuns e muitas vezes você é forçado a trabalhar de acordo com as limitações impostas pela situação.

Nas seções a seguir, discutiremos as etapas necessárias para estabelecer as bases para um entendimento dos requisitos de software — para que o projeto possa ser iniciado de modo que avance na direção de uma solução bem-sucedida.

5.2.1 Identificação de Interessados

Sommerville e Sawyer [Som97] definem “interessados” como “qualquer um que se beneficia de forma direta ou indireta do sistema que está sendo desenvolvido”. Já tivemos a oportunidade de identificar os envolvidos usuais: gerentes de operações, gerentes de produto, pessoal de marketing, clientes internos e externos, usuários finais, consultores, engenheiros de produto, engenheiros de software, engenheiros de suporte e manutenção e outros. Cada interessado tem uma visão diferente do sistema, obtém diferentes benefícios quando o sistema é desenvolvido com êxito e está sujeito a diferentes riscos caso o trabalho de desenvolvimento venha a fracassar.

No início, devemos criar uma lista das pessoas que irão contribuir com sugestões à medida que os requisitos são obtidos (Seção 5.3). A lista inicial crescerá à medida que os interessados forem contatados, pois para cada um deles será feita a pergunta: “Com quem mais você acha que eu deva falar?”.

5.2.2 Reconhecimento de diversos pontos de vista

Pelo fato de existirem muitos interessados diferentes, os requisitos do sistema serão explorados sob vários pontos de vista. Por exemplo, o grupo de marketing está interessado nas funções e recursos que irão suscitar o mercado potencial, facilitando a venda do novo sistema. Os gerentes comerciais estão interessados em um conjunto de recursos que pode ser construído dentro do orçamento e que estarão prontos para atender às oportunidades definidas de ingresso no mercado. Os usuários finais podem querer recursos que sejam familiares a eles e que sejam fáceis de aprender e usar. Os engenheiros de software podem estar preocupados com funções invisíveis aos interessados não técnicos, mas que possibilitam uma infraestrutura que dê suporte a um maior número de funções e recursos comercializáveis. Os engenheiros de suporte talvez enfoquem a facilidade de manutenção do software.

Cada uma dessas partes envolvidas (e outras) contribuirá com informações para o processo de engenharia de requisitos. As informações dos diversos pontos de vista são coletadas, requisitos emergentes talvez sejam inconsistentes ou entrem em conflito uns com os outros. Deve-se classificar as informações de todos os interessados (inclusive os requisitos inconsistentes e conflitantes) de maneira que permita aos tomadores de decisão escolher um conjunto internamente consistente de requisitos para o sistema.

“Coloque três interessados em uma sala e pergunte a eles que tipo de sistema desejam. Provavelmente você obterá quatro ou mais opiniões diferentes.”

Autor desconhecido

⁸ Essa abordagem é altamente recomendada para projetos que adotam uma filosofia de desenvolvimento de software ágil.

5.2.3 Trabalho na busca da colaboração

Se cinco interessados estiverem envolvidos em um projeto de software, talvez tenhamos cinco (ou mais) opiniões diferentes sobre o conjunto de requisitos apropriado. Ao longo dos capítulos anteriores, citei que os clientes (e outros interessados) devem colaborar entre si (evitando insignificantes lutas internas pelo poder) e com os profissionais de engenharia de software, caso se queira obter um sistema bem-sucedido. Mas como a colaboração é atingida?

O trabalho de um engenheiro de requisitos é identificar áreas em comum (requisitos com os quais todos os interessados concordam) e áreas de conflito ou inconsistência (requisitos desejados por um interessado, mas que conflitam com os de outro interessado). É, obviamente, a última categoria que representa um desafio.



Utilização de "pontos de prioridade"

Um modo de resolver requisitos conflitantes e, ao mesmo tempo, entender a importância relativa de todas as necessidades é usar um esquema de "votação" baseado nos pontos de prioridade.

Todos os interessados recebem certo número de pontos de prioridade que podem ser "gastos" em um número qualquer de requisitos. É apresentada uma lista de requisitos, e cada interessado indica a importância relativa de cada um deles (sob o seu

ponto de vista) gastando um ou mais pontos de prioridade nele. Pontos gastos não podem ser reutilizados.

Uma vez que os pontos de prioridade de um interessado tenham se esgotado, nenhuma ação adicional em relação aos requisitos pode ser feita por essa pessoa. O total de pontos dados por todos os interessados a cada requisito dá uma indicação da importância global de cada requisito.

INFORMAÇÕES

Colaboração não significa necessariamente que os requisitos são definidos por um comitê. Em muitos casos, os interessados colaboram dando suas visões dos requisitos, mas um poderoso "campeão dos projetos" (por exemplo, um gerente comercial ou um técnico sênior) pode tomar a decisão final sobre quais os requisitos que passam pelo corte.

5.2.4 Perguntas iniciais

As perguntas feitas na concepção do projeto devem ser "livres de contexto" [Gau89]. O primeiro conjunto de perguntas enfoca o cliente e outros interessados, os benefícios e as metas de projeto globais. Por exemplo, poderíamos perguntar:

- Quem está por trás da solicitação deste trabalho?
- Quem irá usar a solução?
- Qual será o benefício econômico de uma solução bem-sucedida?
- Há uma outra fonte para a solução que você precisa?

Essas perguntas ajudam a identificar todos os interessados no software a ser criado. Além disso, identificam o benefício mensurável de uma implementação bem-sucedida e possíveis alternativas para o desenvolvimento de software personalizado.

O conjunto de perguntas a seguir permite que adquiramos um melhor entendimento do problema e que o cliente expresse suas percepções sobre uma solução:

- Como você caracterizaria uma "boa" saída, que seria gerada por uma solução bem-sucedida?
- Qual(is) problema(s) esta solução irá tratar?
- Você poderia me indicar (ou descrever) o ambiente de negócios em que a solução será usada?
- Restrições ou problemas de desempenho afetam a maneira com que a solução será abordada?

"É melhor conhecer algumas perguntas do que todas as respostas."

James Thurber

? Quais perguntas irão ajudá-lo a obter um entendimento preliminar do problema?

O conjunto final de perguntas concentra-se na eficácia da atividade de comunicação em si. Gause e Weinberg [Gau89] chamam esse conjunto de “metaperguntas” e propõem a seguinte lista (sintetizada):

“Aquele que pergunta é tolo por cinco minutos; aquele que não faz é tolo para sempre.”

Provérbio chinês

- Você é a pessoa correta para responder estas perguntas? Suas respostas são “oficiais”?
- Minhas perguntas são relevantes para o problema que você tem?
- Estaria eu fazendo perguntas demais?
- Alguma outra pessoa poderia me prestar informações adicionais?
- Deveria eu perguntar-lhe algo mais?

Essas (e outras) perguntas irão ajudá-lo a “quebrar o gelo” e iniciar o processo de comunicação essencial para o êxito do levantamento. Porém, uma reunião com formato perguntas-e-respostas não é uma abordagem que tem obtido um sucesso marcante. Na realidade, a sessão Q&A (perguntas e respostas) deveria ser usada apenas no primeiro encontro e depois substituída pelo formato de levantamento de requisitos que combina elementos de resolução de problemas, negociação e especificação. Uma abordagem desse tipo é apresentada na Seção 5.3.

5.3 LEVANTAMENTO DE REQUISITOS

O levantamento de requisitos (também chamado *elicitação de requisitos*) combina elementos de resolução de problemas, elaboração, negociação e especificação. Para encorajar uma abordagem colaborativa e orientada às equipes em relação ao levantamento de requisitos, os interessados trabalham juntos para identificar o problema, propor elementos da solução, negociar diferentes abordagens e especificar um conjunto preliminar de requisitos da solução [Zah90].⁹

5.3.1 Coleta colaborativa de requisitos

Foram propostas várias abordagens para a coleta colaborativa de requisitos. Cada uma delas faz uso de um cenário ligeiramente diferente, porém todas aplicam alguma variação das seguintes diretrizes básicas:

? Quais são as diretrizes básicas para conduzir uma reunião de coleta colaborativa de requisitos?

- As reuniões são conduzidas por e com a participação tanto dos engenheiros de software quanto de outros interessados.
- São estabelecidas regras para preparação e participação.
- É sugerida uma agenda suficientemente formal para cobrir todos os pontos importantes, porém, suficientemente informal para encorajar o fluxo livre de ideias.
- Um “facilitador” (pode ser um cliente, um desenvolvedor ou uma pessoa de fora) dirige a reunião.
- É utilizado um “mecanismo de definições” (que pode ser planilhas, *flip charts*, adesivos de parede ou um boletim eletrônico, salas de bate-papo ou fóruns virtuais).

“Gastamos um bom tempo — a maior parte do esforço de um projeto — não implementando ou testando, mas sim tentando decidir o que construir.”

Brian Lawrence

A meta é identificar o problema, propor elementos da solução, negociar diferentes abordagens e especificar um conjunto preliminar de requisitos da solução em uma atmosfera que seja propícia para o cumprimento da meta. Para melhor compreender o fluxo de eventos à medida que ocorrem, apresento um breve cenário que descreve em linhas gerais a sequência de eventos que levam à reunião para levantamento de requisitos e que acontecem durante e após a reunião.

Durante a concepção (Seção 5.2) perguntas e respostas estabelecem o escopo do problema e a percepção geral de uma solução. Como resultado dessas reuniões iniciais, o desenvolvedor e os clientes redigem uma “solicitação de produto” de uma ou duas páginas.

⁹ Essa abordagem é algumas vezes chamada FAST (*facilitated application specification technique*, ou seja, técnica de especificação de aplicações facilitada).

WebRef

Joint Application Development (JAD) é uma popular técnica para levantamento de requisitos. Uma excelente descrição sobre ela pode ser encontrada em www.carolla.com/wp-jad.htm.



Se um sistema ou produto é atender muitos usuários, esteja absolutamente certo de que os requisitos foram extraídos de uma fonte representativa dos usuários. Se apenas um usuário definiu todas as necessidades, o risco de não aceitação é grande.

"Fatos não cessam de existir por serem ignorados."

Aldous Huxley



Evite o impulso de hostilizar uma sugestão de um cliente classificando-a como "muito cara" ou "pouca prática". O objetivo aqui é negociar uma lista que seja aceitável para todos. Para tanto, você deve ser receptivo a novas ideias.

São escolhidos um local, hora e data para a reunião; é escolhido um facilitador; e os membros da equipe de software e de outros departamentos interessados são convidados a participar. A solicitação de produto é distribuída a todos os participantes antes da data da reunião.

Como exemplo,¹⁰ consideremos um trecho de uma solicitação de produto redigida por uma pessoa de marketing envolvida no projeto *CasaSegura*. Esta pessoa escreve a seguinte narrativa sobre a função de segurança domiciliar que faz parte do *CasaSegura*:

Nossa pesquisa indica que o mercado para sistemas de administração domiciliar está crescendo com taxas de 40% ao ano. A primeira função do *CasaSegura* que lançáramos no mercado seria a função de segurança domiciliar. A maioria das pessoas está familiarizada com "sistemas de alarme", de modo que seria algo fácil de vender.

A função de segurança domiciliar protegeria e/ou reconheceria uma série de "situações" indesejáveis como acesso ilegal, incêndios, inundações, níveis de monóxido de carbono e outras. Ela usará nossos sensores sem fio para detectar cada uma dessas situações. Poderá ser programada pelo proprietário da casa e, automaticamente, ligará para uma agência de monitoramento quando for detectada uma situação destas.

Na realidade, outras pessoas contribuiriam para essa narrativa durante a reunião para levantamento de requisitos e um número consideravelmente maior de informações ficariam disponíveis. Contudo, mesmo com essas informações adicionais, a ambiguidade estaria presente, provavelmente existiriam omissões e poderiam ocorrer erros. Por enquanto, a "descrição funcional" anterior será suficiente.

Ao rever a solicitação de produto nos dias que antecedem a reunião, é pedido a cada participante uma lista de objetos que fazem parte do ambiente que cerca o sistema, outros que devem ser produzidos pelo sistema e aqueles usados pelo sistema para desempenhar suas funções. Além disso, pede-se a cada participante uma outra lista de serviços (processos ou funções) que manipulam ou interagem com os objetos. Por fim, também são desenvolvidas listas de restrições (por exemplo, custo, dimensões, regras comerciais) e de critérios de desempenho (por exemplo, velocidade, precisão). Os participantes são informados que não se espera que as listas sejam exaustivas, mas que reflitam a percepção do sistema de cada pessoa.

Entre os objetos descritos para o *CasaSegura* poderíamos ter o painel de controle, detectores de fumaça, sensores para janelas e portas, detectores de movimento, um alarme, um evento (por exemplo, um sensor foi ativado), um display, um PC, números de telefone, uma ligação telefônica e assim por diante. A lista de serviços poderia incluir *configurar* o sistema, *acionar* o alarme, *monitorar* os sensores, *discar* o telefone, *programar* o painel de controle e *ler* o display (note que os serviços atuam sobre os objetos). De maneira similar, cada participante irá criar listas de restrições (por exemplo, o sistema tem de reconhecer quando os sensores não estão operando, ser amigável, interfacear diretamente com uma linha telefônica comum) e de critérios de desempenho (por exemplo, um evento de sensor seria reconhecido em um intervalo de um segundo e seria implementado um esquema de prioridade para os eventos).

As listas de objetos poderiam ser afixadas nas paredes da sala de reunião utilizando-se grandes folhas de papel, coladas nas paredes por meio de folhas adesivas ou escritas em mural. Como alternativa, poderiam ser postadas em um boletim eletrônico, em um site interno ou colocadas em um ambiente de salas de bate-papo para revisão antes da reunião. De modo ideal, cada entrada deveria ser capaz de ser manipulada separadamente, de modo que as listas pudessem ser combinadas, as entradas modificadas e adições feitas. Nesse estágio, críticas e polêmicas são estritamente proibidas.

¹⁰ Esse exemplo (com extensões e variações) é usado para ilustrar importantes métodos de engenharia de software em vários dos capítulos que vêm a seguir. Como exercício, valeria a pena realizar sua própria reunião para levantamento de requisitos e desenvolver um conjunto de listas para ela.

Depois de as listas individuais terem sido apresentadas, o grupo cria uma lista combinada eliminando entradas redundantes, acrescentando quaisquer ideias novas que surjam durante a discussão, mas não se elimina nada. Segue-se então uma discussão (coordenada pelo facilitador). A lista combinada é reduzida, ampliada ou escrita de outra maneira para refletir apropriadamente o produto/sistema a ser desenvolvido. O objetivo é criar uma lista consensual de objetos, serviços, restrições e desempenho para o sistema a ser construído.

Em muitos casos, um objeto ou serviço descrito em uma lista exigirá maiores explicações. Para tanto, os interessados desenvolvem *miniespecificações* para as entradas das listas.¹¹ Cada miniespecificação é a elaboração de um objeto ou serviço. Por exemplo, a miniespecificação para o objeto **Control Panel** do *CasaSegura* poderia ser:

O painel de controle é a unidade que pode ser montada na parede com tamanho aproximado de 9 x 5 polegadas. O painel de controle possui conectividade sem fio a sensores e a um PC. A interação com os usuários ocorre com um teclado numérico contendo 12 teclas. Um display colorido LCD de 3 x 3 polegadas fornece o feedback do usuário. O software fornece prompts interativos, eco e funções similares.

As miniespecificações são apresentadas a todos os interessados para discussão. Adições, supressões e mais detalhamentos são feitos. Em alguns casos, o desenvolvimento de miniespecificações irá revelar novos objetos, serviços, restrições, ou requisitos de desempenho acrescentados às listas originais. Durante todas as discussões, a equipe pode levantar um assunto que não pode ser resolvido durante a reunião. É mantida uma *lista de questões pendentes* para que essas ideias sejam trabalhadas posteriormente.

CASASEGURA



Condução de uma reunião para levantamento de requisitos

Cena: Uma sala de reunião. A primeira reunião para levantamento de requisitos está em andamento.

Atores: Jamie Lazar, membro da equipe de software; Vinod Raman, membro da equipe de software; Ed Robbins, membro da equipe de software; Doug Miller, gerente da engenharia de software; três membros do Depto. de Marketing; um representante da Engenharia de Produto e um facilitador.

Conversa:

Facilitador (apontando para uma lousa branca): Portanto, esta é a lista atual de objetos e serviços para a função segurança domiciliar.

Representante do Depto. de Marketing:

Segundo o nosso ponto de vista, esta lista cobre aproximadamente todas as funcionalidades.

Vinod: Alguém não mencionou que eles queriam que toda a funcionalidade do CasaSegura pudesse ser acessada via Internet?

Isso incluiria a função de segurança domiciliar, não é mesmo?

Representante do Depto. de Marketing: Sim, você está certo... Teremos de acrescentar essa funcionalidade e os objetos apropriados.

Facilitador: Isso também não acrescentaria algumas restrições?

Jamie: Realmente, restrições técnicas e legais.

Representante da Engenharia de Produto: E isso significa o quê?

Jamie: É melhor termos certeza de que um intruso não conseguirá entrar no sistema, desarmá-lo e roubar o local ou coisa pior. Grande responsabilidade sobre nós.

Doug: Uma grande verdade.

Representante do Depto. de Marketing: Mas ainda assim precisamos que... Apenas ter certeza de impedir que um intruso entre.

Ed: É fácil dizer, o duro é fazer...

Facilitador (interrompendo): Não gostaria de discutir esta questão agora. Anotemos a questão para ser trabalhada no futuro e prossigamos.

(Doug, atuando como o secretário da reunião, faz um apontamento apropriado.)

Facilitador: Tenho a impressão de que ainda há mais coisas a ser consideradas aqui.

(O grupo gasta os 20 minutos seguintes refinando e expandindo os detalhes da função segurança domiciliar.)

¹¹ Em vez de criar uma miniespecificação, muitas equipes de software optam por desenvolver cenários de usuários denominados *casos de uso*. Estes são considerados de forma detalhada na Seção 5.4 e no Capítulo 6.

PONTO-CHAVE

O QFD define necessidades para maximizar a satisfação do cliente.



Todo mundo quer implementar um monte de requisitos fascinantes, porém, tenha cuidado. É assim que o "surgimento incontrolado de novos requisitos" se estabelece. Por outro lado, requisitos fascinantes levam a um produto revolucionário!

WebRef

Outras informações sobre o QFD podem ser obtidas em www.qfdi.org.

5.3.2 Disponibilização da função de qualidade

A disponibilização da função de qualidade (*quality function deployment*, QFD) é uma técnica de gestão da qualidade que traduz as necessidades do cliente para requisitos técnicos do software. O QFD "concentra-se em maximizar a satisfação do cliente por meio do processo de engenharia de software" [Zul92]. Para tanto, enfatiza o entendimento daquilo que é valioso para o cliente e emprega esses valores ao longo do processo de engenharia. O QFD identifica três tipos de necessidades [Zul92]:

Requisitos normais. Refletem os objetivos e metas estabelecidos para um produto ou sistema durante reuniões com o cliente. Se esses requisitos estiverem presentes, o cliente fica satisfeito. Exemplos de requisitos normais poderiam ser tipos de displays gráficos solicitados, funções de sistema específicas e níveis de desempenho definidos.

Requisitos esperados. Esses requisitos estão implícitos no produto ou sistema e podem ser tão fundamentais que o cliente não os declara explicitamente. Sua ausência será causa de grande insatisfação. Exemplos de requisitos esperados são: facilidade na interação homem-máquina, confiabilidade e correção operacional global e facilidade na instalação do software.

Requisitos fascinantes. Esses recursos vão além da expectativa dos clientes e demonstram ser muito satisfatórios quando presentes. Por exemplo, o software para um novo celular vem com recursos-padrão, mas junto vem um conjunto de capacidades não esperadas (por exemplo, tecla multitoque, correio de voz visual) que deleitam todos os usuários do produto.

Embora os conceitos QFD possam ser aplicados ao longo de todo o processo de software [Par 96a], técnicas QFD específicas são aplicáveis à atividade de levantamento de requisitos. O QFD usa observação e entrevistas com clientes, pesquisas e exame de dados históricos (por exemplo, relatórios de problemas) como dados brutos para a atividade de levantamento de requisitos. Esses dados são então traduzidos em uma tabela de requisitos — denominada *tabela da voz do cliente* — revisada com o cliente e outros interessados. Uma série de diagramas, matrizes e métodos de avaliação é então usada para extrair os requisitos esperados e para tentar obter os requisitos fascinantes [Aka04].

5.3.3 Cenários de uso

À medida que os requisitos são levantados, uma visão geral das funções e características começa a se materializar. Entretanto, é difícil progredir para atividades de engenharia de software mais técnicas até que entendamos como tais funções e características serão usadas por diferentes classes de usuários. Para tanto, os desenvolvedores e usuários podem criar um conjunto de cenários que identifique um roteiro de uso para o sistema a ser construído. Os cenários, normalmente chamados *casos de uso* [Jac92], fornecem uma descrição de como o sistema será utilizado. Casos de uso são discutidos de forma mais detalhada na Seção 5.4.

CASA SEGURA**Desenvolvimento de um cenário de uso preliminar**

Cena: Sala de reuniões, na qual prossegue a primeira reunião de levantamento de requisitos.

Atores: Jamie Lazar, membro da equipe de software; Vinod Raman, membro da equipe de software; Ed Robbins, membro da equipe de software; Doug Miller, gerente da engenharia de soft-

ware; três membros do Depto. de Marketing; um representante da Engenharia de Produto e um facilitador.

Conversa:

Facilitador: Andamos conversando sobre segurança de acesso à funcionalidade CasaSegura que poderá ser acessada via Internet. Gostaria de tentar algo. Vamos desenvolver um cenário de uso para acesso à função segurança domiciliar.

Jamie: Como?

Facilitador: Podemos realizar isso de várias maneiras, mas, por enquanto, gostaria de manter as coisas realmente informais. Conte-nos (ele aponta para um representante do Depto. de Marketing) como você imagina o acesso ao sistema.

Representante do Depto. de Marketing: Huum... Bem, esse é o tipo de coisa que eu faria se estivesse fora e tivesse que deixar alguém em casa, digamos uma empregada ou um encanador, que não teriam o código de acesso.

Facilitador (sorrindo): Isso que você falou é a razão para você fazê-lo... Diga-me como realmente faria isso.

Representante do Depto. de Marketing: Huum... A primeira coisa de que precisaria seria um PC. Entraria em um site que deveríamos manter para todos os usuários da CasaSegura. Forneceria meu nome de usuário e...

Vinod (interrompendo): A página teria de ser segura e criptografada para garantir que estamos seguros e...

Facilitador (interrompendo): Essas são informações adequadas, Vinod, porém são técnicas. Concentremo-nos apenas em como o usuário final usará essa capacidade. OK?

Vinod: Sem problemas.

Representante do Depto. de Marketing: Bem, como estava dizendo, entraria em um site e forneceria meu nome de usuário e dois níveis de senhas.

Jamie: O que acontece se eu esquecer minha senha?

Facilitador (interrompendo): Excelente observação, Jamie, mas não tratemos disso agora. Faremos um registro de sua observação e a chamaremos de exceção. Tenho certeza de que existirão outras.

Representante do Depto. de Marketing: Após eu introduzir as senhas, uma tela representando todas as funções da CasaSegura aparecerá. Eu selecionaria a função de segurança domiciliar. O sistema poderia solicitar que eu verificasse quem sou eu, digamos, perguntando meu endereço ou telefone ou algo do gênero. Em seguida, ele exibiria uma imagem do painel de controle do sistema de segurança com uma lista das funções que eu poderia executar — armar o sistema, desarmá-lo, desarmar um ou mais sensores. Suponho que ele também me permitiria reconfigurar zonas de segurança e outros itens similares, mas, não tenho certeza disso.

[Enquanto o representante do Depto. de Marketing continua falando, Doug faz anotações de maneira ininterrupta; essas formam a base para o primeiro cenário de uso informal. Como alternativa, poderia ser solicitado ao representante do Depto. de Marketing que escrevesse o cenário, mas isso seria feito fora da reunião.]

5.3.4 Artefatos do levantamento de requisitos

Os artefatos produzidos como consequência do levantamento de requisitos variarão dependendo do tamanho do sistema ou produto a ser construído. Para a maioria dos sistemas, entre os artefatos, temos:

? Quais informações são produzidas como consequência do levantamento de requisitos?

- Uma declaração da necessidade e viabilidade.
- Uma declaração restrita do escopo para o sistema ou produto.
- Uma lista de clientes, usuários e outros interessados que participaram do levantamento de requisitos.
- Uma descrição do ambiente técnico do sistema.
- Uma lista de requisitos (preferencialmente organizada por função) e as restrições de domínio que se aplicam a cada uma delas.
- Um conjunto de cenários de uso que esclarecem o uso do sistema ou produto sob diferentes condições operacionais.
- Quaisquer protótipos desenvolvidos para melhor definição dos requisitos.

Cada um desses artefatos é revisado por todas as pessoas que participaram do levantamento de requisitos.

5.4 DESENVOLVIMENTO DE CASOS DE USO

Em um livro que discute como redigir casos de uso eficazes, Alistair Cockburn [Coc01b] observa que “um caso de uso captura um contrato... [que] descreve o comportamento do sistema sob várias condições à medida que o sistema responde a uma solicitação de um de seus interessados...”. Essencialmente, um caso de uso conta uma história estilizada sobre como um usuário final (desempenhando um de uma série de papéis possíveis) interage com o sistema sob um

PONTO-CHAVE

Os casos de uso são definidos sob o ponto de vista de um ator. Ator é um papel que as pessoas (usuários) ou dispositivos desempenham à medida que interagem com o software.

WebRef

Um excelente artigo sobre casos de uso pode ser baixado de www.ibm.com/developerworks/webservices/library/codesign7.html.

O que preciso saber para desenvolver um caso de uso efetivo?

conjunto de circunstâncias específicas. A história poderia ser um texto narrativo, uma descrição geral das tarefas ou interações, uma descrição baseada em gabaritos ou uma representação esquemática. Independentemente de sua forma, um caso de uso representa o software ou o sistema do ponto de vista do usuário final.

O primeiro passo ao escrever um caso de uso é definir o conjunto de "atores" envolvidos na história. *Atores* são as diferentes pessoas (ou dispositivos) que usam o sistema ou produto no contexto da função e comportamento a ser descritos. Os atores representam os papéis que pessoas (ou dispositivos) desempenham enquanto o sistema opera. Definido de maneira um pouco mais formal, ator é qualquer coisa que se comunica com o sistema ou o produto e que é externa ao sistema em si. Todo ator possui uma ou mais metas ao usar o sistema.

É importante notar que o ator e o usuário final não são necessariamente a mesma coisa. O usuário típico poderia desempenhar uma série de papéis diferentes ao usar um sistema, ao passo que o ator representa uma classe de entidades externas (normalmente, mas não sempre, pessoas) que desempenham apenas um papel no contexto do caso de uso. Como exemplo, consideremos um operador de máquina (um usuário) que interage com o computador de controle de uma célula de fabricação contendo uma série de robôs e máquinas comandada por controle numérico. Após uma revisão cuidadosa dos requisitos, o software para o computador de controle requer quatro modos (papéis) diferentes para interação: modo de programação, modo de teste, modo de monitoramento e modo de diagnóstico. Portanto, podem ser definidos quatro atores: programador, testador, monitorador e diagnosticador. Em alguns casos, o operador de máquina pode desempenhar todos esses papéis. Em outros, pessoas diferentes poderiam desempenhar o papel de cada ator.

Pelo fato de o levantamento de requisitos ser uma atividade evolucionária, nem todos os atores são identificados durante a primeira iteração. É possível identificar atores primários [Jac92] durante a primeira iteração e atores secundários quando mais fatos são aprendidos sobre o sistema. Os *primários* interagem para atingir a função necessária do sistema e obter o benefício desejado do sistema. Eles trabalham direta e frequentemente com o software. Os *secundários* dão suporte ao sistema, de modo que os primários possam realizar seu trabalho.

Uma vez que os atores tenham sido identificados, os casos de uso podem ser desenvolvidos. Jacobson [Jac92] sugere uma série de perguntas¹² que devem ser respondidas por um caso de uso:

- Quem é(são) o(s) ator(es) primário(s) e o(s) ator(es) secundário(s)?
- Quais são as metas do ator?
- Que precondições devem existir antes de uma história começar?
- Que tarefas ou funções principais são realizadas pelo ator?
- Que exceções deveriam ser consideradas à medida que uma história é descrita?
- Quais são as variações possíveis na interação do ator?
- Que informações de sistema o ator adquire, produz ou modifica?
- O ator terá de informar o sistema sobre mudanças no ambiente externo?
- Que informações o ator deseja do sistema?
- O ator gostaria de ser informado sobre mudanças inesperadas?

Relembrando as necessidades básicas do *CasaSegura*, definimos quatro atores: **proprietário** (um usuário), **gerente de ativação** (provavelmente a mesma pessoa que o **proprietário**, porém desempenhando um papel diferente), **sensores** (dispositivos conectados ao sistema) e o **subsistema de monitoramento e resposta** (a estação central que monitora a função segurança domiciliar do *CasaSegura*). Para o propósito deste exemplo, consideraremos apenas o ator **proprietário**, que interage com a função segurança domiciliar de uma de várias maneiras diferentes usando o painel de controle de alarme ou um PC:

¹² As perguntas de Jacobson foram estendidas para dar uma visão mais completa do conteúdo dos casos de uso.

- Digita uma senha para permitir todas as demais interações.
- Consulta o estado de uma zona de segurança.
- Consulta o estado de um sensor.
- Pressiona o botão de alarme em caso de emergência.
- Ativa/desativa o sistema de segurança.

Considerando-se a situação em que o proprietário do imóvel usa o painel de controle, o caso de uso básico para ativação do sistema é o seguinte:¹³

1. O proprietário observa o painel de controle do *CasaSegura* (Figure 5.1) para determinar se o sistema está pronto para entrada. Se o sistema não estiver pronto, será mostrada uma mensagem *não disponível* no display LCD e o proprietário terá de fechar manualmente as janelas ou portas para que a mensagem *não disponível* desapareça. [Uma mensagem *não disponível* implica que um sensor está aberto; isto é, que uma porta ou janela está aberta.]
2. O proprietário usa o teclado numérico para introduzir uma senha de quatro dígitos. A senha é comparada com a senha válida armazenada no sistema. Se estiver incorreta, o painel de controle emitirá um bipe uma vez e se autorreinciará na espera de entrada adicional. Se a senha estiver correta, o painel de controle aguarda por novas ações.
3. O proprietário seleciona e digita *em casa* ou *fora de casa* (veja a Figura 5.1) para ativar o sistema. *Em casa* ativa apenas sensores periféricos (os sensores para detecção de movimento interno são desativados). *Fora de casa* ativa todos os sensores.
4. Quando ocorre a ativação, uma luz de alarme vermelha pode ser observada pelo proprietário.



Os casos de uso são normalmente escritos de forma informal. Entretanto, use o modelo aqui mostrado para garantir que você tratou todas as questões chave.

O caso de uso básico apresenta uma história detalhada que descreve a interação entre o ator e o sistema.

Em muitas ocasiões, os casos de uso são mais elaborados para dar um nível de detalhes consideravelmente maior sobre a interação. Por exemplo, Cockburn [Coc01b] sugere o seguinte modelo para descrições detalhadas de casos de uso:

Caso de uso:	<i>IniciarMonitoramento</i>
Ator primário:	Proprietário.
Meta do contexto:	Ativar o sistema para monitoramento dos sensores quando o proprietário deixa a casa ou nela permanece.

FIGURA 5.1

Painel de controle do CasaSegura



¹³ Note que esse caso de uso difere da situação em que o sistema é acessado via Internet. Nessa situação, a interação ocorre através do painel de controle e não da interface gráfica do usuário (GUI) fornecida quando é utilizado um PC.

Precondições: O sistema foi programado para uma senha e para reconhecer vários sensores.

Disparador: O proprietário decide "acionar" o sistema, isto é, ativar as funções de alarme.

Cenário:

1. Proprietário: observa o painel de controle
2. Proprietário: introduz a senha
3. Proprietário: seleciona "em casa" ou "fora de casa"
4. Proprietário: observa a luz de alarme vermelha para indicar que o CasaSegura foi armado

Exceções:

1. O painel de controle encontra-se no estado *não disponível*: o proprietário verifica todos os sensores para determinar quais estão abertos; fechando-os.
2. A senha incorreta (o painel de controle emite um bipe): o proprietário introduz novamente a senha, desta vez correta.
3. Senha não reconhecida: o subsistema de monitoramento e resposta deve ser contatado para reprogramar a senha.
4. É selecionado *em casa*: o painel de controle emite dois bipes e uma luz de *em casa* é acesa; os sensores periféricos são ativados.
5. É selecionado *fora de casa*: o painel de controle emite três bipes e uma luz *fora de casa* é acesa; todos os sensores são ativados.

Prioridade: Essencial, deve ser implementada

Quando disponível: Primeiro incremento

Frequência de uso: Várias vezes por dia

Canal com o ator: Via interface do painel de controle

Atores secundários: Técnico de suporte, sensores

Canais com os atores secundários:

Técnico de suporte: linha telefônica

Sensores: interfaces *hardwired* e de radiofrequência

Questões em aberto:

1. Existiria um modo de ativar o sistema sem o uso de uma senha ou com uma senha abreviada?
2. Deveria o painel de controle exibir outras mensagens de texto?
3. Quanto tempo o proprietário tem para introduzir a senha a partir do instante em que a primeira tecla é pressionada?
4. Existe alguma maneira de desativar o sistema antes de ser realmente ativado?

Casos de uso para outras interações do **proprietário** seriam desenvolvidos de maneira similar. É importante revisar cada caso com cuidado. Se algum elemento da interação for ambíguo, é provável que uma revisão do caso de uso irá indicar um problema.

CASA SEGURA**Desenvolvimento de um diagrama de caso de uso detalhado**

Cena: Sala de reuniões, na qual prossegue a reunião para levantamento de necessidades.

Atores: Jamie Lazar, membro da equipe de software; Vinod Raman, membro da equipe de software; Ed Robbins, membro da

equipe de software; Doug Miller, gerente da engenharia de software; três membros do Depto. de Marketing; um representante da Engenharia de Produto e um facilitador.

Conversa:

Facilitador: Investimos um tempo razoável falando sobre a funcionalidade de segurança domiciliar do CasaSegura. Durante o intervalo esbocei um diagrama de caso de uso para sintetizar

os cenários importantes que fazem parte desta função. Deem uma olhada.

[Todos os participantes observam a Figura 5.2.]

Jamiet: Estou apenas começando a aprender a notação UML.¹⁴ Portanto, a função de segurança domiciliar é representada pelo retângulo grande com as elipses em seu interior? E as elipses representam os casos de uso que redigimos?

Facilitador: Certo. E as figuras de bonecos representam atores — as pessoas ou coisas que interagem com o sistema conforme descrito pelo caso de uso... Oh, eu uso o quadrado legendado para representar um ator que não é uma pessoa... Neste caso, sensores.

Doug: Isso é permitido na UML?

Facilitador: Permissão não é o problema. O ponto é comunicar a informação. Eu vejo o uso de uma figura de boneco para representar um dispositivo enganoso. Portanto, adaptei um pouco as coisas. Não acho que isso irá criar problemas.

Vinod: OK, temos narrativas de casos de uso para cada uma das elipses. Precisamos desenvolver narrativas baseadas nos modelos detalhados sobre os quais li a respeito?

Facilitador: Provavelmente, mas isso pode esperar até que tenhamos considerado outras funções do CasaSegura.

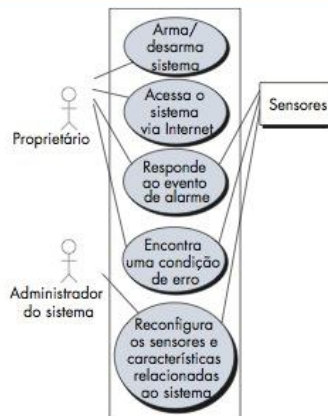
Representante do Depto. de Marketing: Espere um pouco. Fiquei observando este diagrama e de repente me dei conta de que deixamos passar algo.

Facilitador: Ah é? Diga-me o que deixamos passar.

[A reunião continua.]

FIGURA 5.2

Diagrama de caso de uso em UML para a função de segurança domiciliar do CasaSegura



Desenvolvimento de caso de uso

Objetivo: Auxiliar no desenvolvimento de casos de uso através de modelos e mecanismos automatizados para avaliar a clareza e a consistência.

Mecânica: A mecânica das ferramentas varia. Em geral, as ferramentas de caso de uso fornecem modelos em que se pode preencher os espaços em branco para criar casos de uso efetivos. A maior parte da funcionalidade de caso de uso está embutida em um conjunto de funções de engenharia de requisitos mais abrangentes.

FERRAMENTAS DO SOFTWARE

Ferramentas representativas:¹⁵

A grande maioria das ferramentas de modelagem de análise baseadas em UML oferece recursos de texto como gráficos para o desenvolvimento e a modelagem de casos de uso.

Objects by Design

(www.objectsbydesign.com/tools/umltools_byCompany.html) oferece um grande número de links para ferramentas desse tipo.

¹⁴ Um breve tutorial sobre a UML é apresentado no Apêndice 1 para aqueles que não estão familiarizados com sua notação.

¹⁵ As ferramentas aqui apresentadas não significam um aval, mas sim, uma amostra de ferramentas nessa categoria. Na maioria dos casos, seus nomes são marcas registradas pelos respectivos desenvolvedores.

5.5 CONSTRUÇÃO DO MODELO DE ANÁLISE¹⁶

O intuito do modelo de análise é fornecer uma descrição dos domínios de informação, funcional e comportamental necessários para um sistema baseado em computadores. O modelo se modifica dinamicamente à medida que você aprende mais sobre o sistema a ser construído e outros interessados adquirem um melhor entendimento sobre aquilo que eles realmente querem. Por essa razão, o modelo de análise é uma reprodução das necessidades em determinado momento. Você deve esperar que ele sofra mudanças.

À medida que o modelo de requisitos evolui, certos elementos se tornarão relativamente estáveis, fornecendo uma sólida base para as tarefas posteriores do projeto. Entretanto, outros elementos do modelo podem ser mais voláteis, indicando que os interessados ainda não têm um entendimento completo das necessidades para o sistema. O modelo de análise e os métodos usados para construí-lo são apresentados em detalhe nos Capítulos 6 e 7. Apresento uma breve visão geral nas seções a seguir.

5.5.1 Elementos do modelo de análise

Há várias maneiras de examinar os requisitos para um sistema baseado em computador. Alguns profissionais de software argumentam que é melhor selecionar um modo de representação (por exemplo, o caso de uso) e aplicá-lo em detrimento de todos os demais. Outros profissionais acreditam que vale a pena usar uma série de modos de representação para representar o modelo de requisitos. Modos de representação diferentes nos forçam a considerar as necessidades de diferentes pontos de vista — uma abordagem com maior probabilidade de revelar omissões, inconsistências e ambiguidades.

Os elementos específicos do modelo de análise são ditados pelo método de modelagem de análise (Capítulos 6 e 7) que será usado. Entretanto, um conjunto de elementos genéricos é comum à maioria dos modelos de análise.

Elementos baseados em cenários. O sistema é descrito sob o ponto de vista do usuário usando uma abordagem baseada em cenários. Por exemplo, casos de uso básicos (Seção 5.4) e seus diagramas de casos de uso correspondentes (Figura 5.2) evolui para casos de uso mais elaborados baseados em modelos. Elementos do modelo de requisitos baseado em cenários são em geral a primeira parte do modelo a ser desenvolvida. Como tal, servem como entrada para a criação de outros elementos de modelagem. A Figura 5.3 mostra um diagrama¹⁷ de atividades em UML para o levantamento de requisitos e os representa utilizando casos de uso. São mostrados três níveis da elaboração, culminando em uma representação baseada em cenários.

Elementos baseados em classes. Cada cenário de uso implica um conjunto de objetos manipulados à medida que um ator interage com o sistema. Esses objetos são categorizados em classes — um conjunto de coisas que possuem atributos similares e comportamentos comuns. Por exemplo, um diagrama de classes UML pode ser utilizado para representar uma classe **Sensor** para a função de segurança do *CasaSegura* (Figura 5.4). Note que o diagrama enumera os atributos dos sensores (por exemplo, nome, tipo) e as operações (por exemplo, *identificar*, *habilitar*) que podem ser aplicadas para modificar tais atributos. Além dos diagramas de classes, outros elementos de modelagem de análise descrevem o modo pelo qual as classes colaboram entre si e os relacionamentos e interações entre as classes. Estes são discutidos de forma mais detalhada no Capítulo 7.

Elementos comportamentais. O comportamento de um sistema baseado em computadores pode ter um efeito profundo sobre o projeto escolhido e a abordagem de implementação



É sempre uma boa ideia fazer com que os interessados se envolvam. Uma das melhores formas para tal é pedir a cada interessado que escreva casos de uso que descrevam como o software será utilizado.



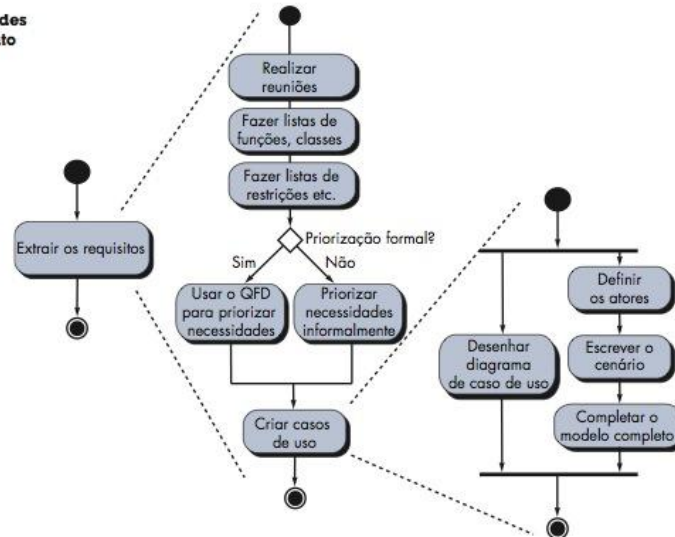
Uma maneira de isolar classes é procurar substantivos descritivos em um texto de caso de uso. Pelo menos alguns dos substantivos serão candidatos a classes. Mais sobre isso no Capítulo 8.

¹⁶ Ao longo deste livro, uso os termos *modelo de análise* e *modelo de requisito* como sinônimos. Ambos se referem às representações dos domínios de informação, funcional e comportamental que descrevem as necessidades dos problemas.

¹⁷ Um breve tutorial sobre a UML é apresentado no Apêndice 1 para aqueles que não estão familiarizados com sua notação.

FIGURA 5.3

Diagramas de atividades UML para levantamento de requisitos



aplicada. Portanto, o modelo de análise deve fornecer elementos de modelagem que descrevem comportamento.

O *diagrama de estados* é um método para representar o comportamento de um sistema através da representação de seus estados e dos eventos que fazem com que o sistema mude de estado. *Estado* é qualquer modo de comportamento externamente observável. Além disso, o diagrama de estados indica ações (por exemplo, ativação de processos) tomadas em decorrência de determinado evento.

Para ilustrar o uso de um diagrama de estados, considere o software embutido no painel de controle do *CasaSegura* responsável pela leitura das entradas feitas pelos usuários. A Figura 5.5 mostra um diagrama de estados UML simplificado.

Além das representações comportamentais do sistema como um todo, o comportamento de classes individuais também pode ser modelado. Uma discussão mais aprofundada de modelagem comportamental é apresentada no Capítulo 7.

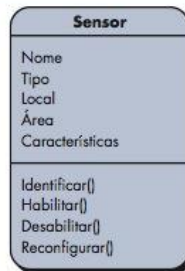
Elementos orientados a fluxos. Informações são transformadas à medida que fluem através de um sistema baseado em computador. O sistema aceita entrada em uma variedade de formas, aplica funções para transformá-las e gera saída também em uma variedade de formas. A entrada pode ser um sinal de controle transmitido por um transdutor, uma série de números digitados por um operador, um pacote de informações transmitido em um link de rede ou um arquivo de dados volumoso recuperado de armazenamento secundário. A(s) transformação(ões) pode(m) compreender desde uma única comparação lógica, um algoritmo numérico complexo até uma abordagem por inferência de regras de um sistema especialista. A saída poderia acender um único LED ou gerar um relatório de 200 páginas. Na realidade, podemos criar um modelo de fluxos para qualquer sistema baseado em computadores, indiferentemente de seu tamanho e complexidade. Uma discussão mais detalhada da modelagem de fluxos é apresentada no Capítulo 7.

PONTO-CHAVE

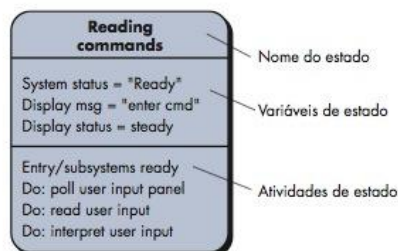
Um estado é um modo de comportamento externamente observável. Estímulos externos provocam transições entre estados.

FIGURA 5.4

Diagrama de classes para Sensor

**FIGURA 5.5**

Notação de um diagrama de estados UML



CASA SEGURA



Modelagem comportamental inicial

Cena: Sala de reuniões, na qual prossegue a primeira reunião para levantamento de requisitos.

Atores: Jamie Lazar, membro da equipe de software; Vinod Raman, membro da equipe de software; Ed Robbins, membro da equipe de software; Doug Miller, gerente da engenharia de software; três membros do Depto. de Marketing; um representante da Engenharia de Produto e um facilitador.

Conversa:

Facilitador: Estamos prestes a finalizar nossa discussão sobre a funcionalidade segurança domiciliar do CasaSegura. Antes de fazê-lo, gostaria de discutir o comportamento da função.

Representante do Depto. de Marketing: Não entendi o que você quis dizer com comportamento.

Ed (sorrindo): Trata-se de dar ao produto um "tempo de espera" caso ele se comporte mal.

Facilitador: Não exatamente. Permita-me explicar.

(O facilitador explica os fundamentos da modelagem comportamental à equipe de levantamento de requisitos.)

Representante do Depto. de Marketing: Isso me parece um tanto técnico. Não estou certo se serei de alguma ajuda aqui.

Facilitador: Certamente que você pode ajudar. Que comportamento você observa do ponto de vista de usuário?

Representante do Depto. de Marketing: Bem, o sistema estará monitorando os sensores. Lendo comandos do proprietário. Mostrando seu estado.

Facilitador: Viu, você pode fazê-lo.

Jamie: Ele também irá indagar o PC para determinar se há qualquer entrada dele proveniente, por exemplo, acesso baseado em Internet ou informações de configuração.

Vinod: Sim, de fato, configurar o sistema é um estado em si.

Doug: Pessoal, vocês estão enrolando.

Pensemos um pouco mais... Existe uma maneira de colocar esta coisa em um diagrama?

Facilitador: Existe, mas adiemos para logo depois da reunião.

5.5.2 Padrões de análise

Qualquer um que tenha feito engenharia de requisitos em mais do que uns poucos projetos de software começa a perceber a recorrência de certos problemas ao longo de todos os projetos em um campo de aplicação específico.¹⁸ Tais *padrões de análise* [Fow97] sugerem soluções (por exemplo, uma classe, função ou comportamento) no campo de aplicação que pode ser reutilizada na modelagem de muitas aplicações.

Geyer-Schulz e Hahsler [Gey01] sugerem dois benefícios que podem estar associados ao uso de padrões de análise:

Primeiro, os padrões de análise aceleram o desenvolvimento de modelos de análise abstratos que capturam os principais requisitos do problema concreto, fornecendo modelos de análise reutilizáveis com exemplos, bem como uma descrição de vantagens e limitações. Em segundo lugar, os padrões de análise facilitam a transformação do modelo de análise em um modelo de projeto, sugerindo padrões de projeto e soluções confiáveis para problemas comuns.

Os padrões de análise são integrados ao modelo de análise por meio de referência ao nome do padrão. Eles também são armazenados em um repositório de modo que os engenheiros de requisitos podem usar recursos de busca para encontrá-los e aplicá-los. Informações sobre um padrão de análise (e outros tipos de padrões) são apresentadas em um modelo padrão [Gey01]¹⁹ discutido de maneira mais detalhada no Capítulo 12. Exemplos de padrões de análise e uma discussão mais ampla deste tópico são apresentados no Capítulo 7.

5.6 NEGOCIAÇÃO DE REQUISITOS

“Compromisso é a arte de dividir um bolo de tal forma que cada um acredite que ficou com o pedaço maior.”

Ludwig Erhard

WebRef

Um breve artigo sobre negociação para requisitos de software pode ser baixado de www.alexandregyad.com/publications/Software_Requirements_Negotiation_Some_Lessons_Learned.html.

Em um contexto de engenharia de requisitos ideal, as tarefas de início, levantamento e elaboração determinam os requisitos do cliente com detalhes suficientes para prosseguir para atividades de engenharia de software subsequentes. Infelizmente, isso raramente acontece. Na realidade, talvez você tenha de iniciar uma negociação com um ou mais interessados. Na maioria dos casos, solicita-se aos interessados contrabalançar funcionalidade, desempenho e outras características do produto ou sistema, em função do custo e tempo para chegar ao mercado. O intuito da negociação é desenvolver um plano de projeto que atenda às necessidades dos interessados e, ao mesmo tempo, reflita as restrições do mundo real (por exemplo, tempo, pessoal, orçamento) impostas à equipe de software.

As melhores negociações buscam ao máximo um resultado “ganha-ganha”.²⁰ Ou seja, os interessados ganham obtendo um sistema ou produto que satisfaz a maioria de suas necessidades e você (como membro da equipe de software) ganha trabalhando com prazos de entrega e orçamentos reais e atingíveis.

Boehm [Boe98] define um conjunto de atividades de negociação no início de cada iteração do processo de software. Mais do que uma simples atividade de comunicação com o cliente, são definidas as seguintes atividades:

1. Identificação dos principais interessados no sistema ou subsistema.
2. Determinação das “condições de ganho” dos interessados.
3. Negociação das condições de ganho dos interessados para reconciliá-las em um conjunto de condições “ganha-ganha” para todos os envolvidos (inclusive a equipe de software).

O êxito na concretização dessas etapas atinge um resultado em que todos saem ganhando, critério-chave para prosseguir para atividades de engenharia de software subsequentes.

¹⁸ Em alguns casos, essa recorrência de problemas acontece independentemente do campo de aplicação. Por exemplo, as características e funções usadas para resolver problemas de interfaces do usuário são comuns independentemente do campo de aplicação considerado.

¹⁹ Uma grande variedade de modelos de padrões foi proposta na literatura. Caso tenha interesse, consulte [Fow97], [Gam95], [Yac03] e [Bus07] entre as muitas fontes.

²⁰ Foram escritos dezenas de livros sobre habilidades de negociação (por exemplo, [Lew06], [Rai06], [Fis06]). É uma das mais importantes habilidades que você pode aprender. Leia um deles.



A arte da negociação

Aprender a negociar eficazmente pode ser bem útil para toda a sua vida técnica e pessoal. Vale muito considerar as seguintes diretrizes:

1. *Reconhecer que não se trata de uma competição.* Para ser bem-sucedido, ambas as partes têm de ter a sensação de que ganharam ou atingiram algum objetivo. Ambas terão de ceder.
2. *Planejar uma estratégia.* Decidir o que você quer atingir; o que a outra parte quer atingir e como você irá fazer para que ambas aconteçam.
3. *Ouvir ativamente.* Não trabalhe na formulação de sua resposta enquanto a outra parte estiver falando. Ouça-a. É provável que você tome conhecimento de algo que irá ajudá-lo a negociar melhor sua posição.
4. *Concentrar-se nos interesses da outra parte.* Não assuma posições rígidas caso queira evitar conflitos.
5. *Não deixar a negociação ir para o lado pessoal.* Concentre-se no problema que precisa ser resolvido.
6. *Seja criativo.* Não tenha medo de inovar caso se encontre em um impasse.
7. *Estar preparado para se comprometer.* Uma vez que se tenha chegado a um acordo, seja objetivo; comprometa-se e vá adiante.

INFORMAÇÕES

CASA SEGURA



O início de uma negociação

Cena: Sala da Lisa Perez, após a primeira reunião para levantamento de requisitos.

Atores: Doug Miller, gerente de engenharia de software e Lisa Perez, gerente de marketing.

Conversa:

Lisa: Bem, ouvi dizer que a primeira reunião transcorreu muito bem.

Doug: Na verdade, foi assim mesmo. Você mandou bons representantes do seu departamento para a reunião... Eles realmente contribuíram.

Lisa (sorrindo): É, na verdade eles me contaram que se engajaram no processo e que não foi uma "atividade de torrar os miolos".

Doug (rindo): Tomarei cuidado para me despojar do jargão técnico na próxima vez que eu visitar... Veja, Lisa, acho que vamos ter problemas para entregar toda a funcionalidade para o sistema de segurança domiciliar nas datas que sua gerência está propondo. Eu sei, é prematuro, porém já andei fazendo um planejamento preliminar e...

Lisa (franzindo a testa): Temos de ter isso até aquela data, Doug. De que funcionalidade você está falando?

Doug: Presumo que consigamos ter a funcionalidade de segurança domiciliar completa até a data-limite, mas teremos de postergar o acesso via Internet para a segunda versão.

Lisa: Doug, é o acesso via Internet que dá todo o charme ao CasaSegura. Vamos criar toda a campanha de marketing em torno disso. Temos de ter esse acesso!

Doug: Entendo sua situação, realmente. O problema é que, para lhes dar o acesso via Internet, teremos de construir e ter funcionando um site totalmente seguro. Isso demanda tempo e pessoal. Também teremos de agregar um monte de outras funções na primeira versão... Não acredito que possamos fazer isso com os recursos que temos no momento.

Lisa (ainda franzindo a testa): Entendo, mas temos de descobrir uma maneira de ter tudo isso pronto. É crítico para as funções de segurança domiciliar e para outras funções também... Estas últimas poderão esperar até a próxima versão... Concordo com isso.

Lisa e Doug parecem ter chegado a um impasse, mas eles têm de negociar uma solução para o problema. Poderiam os dois "ganhar" neste caso? Fazendo o papel de mediador, o que você sugeriria?

5.7 VALIDAÇÃO DOS REQUISITOS

À medida que cada elemento do modelo de requisitos é criado, é examinado em termos de inconsistência, omissões e ambiguidade. Os requisitos representados pelo modelo são priorizados pelos interessados e agrupados em pacotes de requisitos que serão implementados como incrementos de software. Uma revisão do modelo de requisitos trata as seguintes questões:

? Ao revisar os requisitos, que perguntas devo fazer?

- Cada um dos requisitos é consistente com os objetivos globais para o sistema/produto?
- Todos os requisitos foram especificados no nível de abstração apropriado? Ou seja, algum dos requisitos fornece um nível de detalhe técnico inapropriado no atual estágio?

- O requisito é realmente necessário ou representa um recurso adicional que talvez não seja essencial para o objetivo do sistema?
- Cada um dos requisitos é limitado e sem ambiguidade?
- Cada um dos requisitos possui atribuição? Ou seja, uma fonte (em geral, um indivíduo específico) é indicada para cada requisito?
- Algum dos requisitos conflita com outros requisitos?
- Cada um dos requisitos é atingível no ambiente técnico que irá abrigar o sistema ou produto?
- Cada um dos requisitos pode ser testado, uma vez implementado?
- O modelo de requisitos reflete, de forma apropriada, a informação, função e comportamento do sistema a ser construído?
- O modelo de requisitos foi “dividido” para expor progressivamente informações mais detalhadas sobre o sistema?
- Os padrões de requisitos foram utilizados para simplificar o modelo de requisitos? Todos os padrões foram validados adequadamente? Todos os padrões são consistentes com os requisitos do cliente?

Essas e outras perguntas devem ser feitas e respondidas para garantir que o modelo de requisitos reflita de maneira precisa as necessidades do interessado e forneça uma base sólida para o projeto.

5.8 RESUMO

As tarefas da engenharia de requisitos são conduzidas para estabelecer uma base sólida para o projeto e a construção. A engenharia de requisitos ocorre durante as atividades de comunicação com o cliente e de modelagem que são definidas para o processo genérico de software. São conduzidas sete funções distintas de engenharia de requisitos — concepção, levantamento, elaboração, negociação, especificação, validação e gestão — pelos membros da equipe de software.

Na concepção do projeto, os interessados estabelecem os requisitos básicos do problema, definem restrições de projeto primordiais e tratam as principais características e funções que têm de estar presentes para que o sistema atenda seus objetivos. Essas informações são refinadas e expandidas durante o levantamento — uma atividade de levantamento de requisitos que faz uso de reuniões com a participação de um facilitador, do QFD e do desenvolvimento de cenários de uso.

A elaboração expande ainda mais as necessidades em um modelo — um conjunto de elementos comportamentais, orientados a fluxos e baseados em cenários e classes. O modelo pode fazer referência a padrões de análise, soluções para problemas de análise recorrentes em diferentes aplicações.

À medida que os requisitos são identificados e o modelo de análise é criado, a equipe de software e outros interessados no projeto negociam a prioridade, disponibilidade e custo relativo de cada requisito. O intuito dessa negociação é desenvolver um plano de projeto realista. Além disso, cada requisito e o modelo de análise como um todo é validado em relação às necessidades do cliente para garantir que será construído o sistema correto.

PROBLEMAS E PONTOS A PONDERAR

5.1. Por que um número muito grande de desenvolvedores de software não dedica muita atenção à engenharia de requisitos? Existiria alguma circunstância em que poderíamos deixá-la de lado?

5.2. Foi lhe dada a responsabilidade de extrair os requisitos de um cliente que lhe diz que está muito ocupado para poder atendê-lo. O que você deveria fazer?

- 5.3.** Discuta alguns dos problemas que ocorrem quando os requisitos têm de ser obtidos de três ou quatro clientes diferentes.
- 5.4.** Por que dizemos que o modelo de análise representa uma reprodução de um sistema em determinado momento?
- 5.5.** Suponhamos que você tenha convencido o cliente (você é um excelente vendedor) a concordar com todas as suas exigências como desenvolvedor. Isso o torna um mestre da negociação? Por quê?
- 5.6.** Desenvolva pelo menos três "perguntas livres de contexto" que você faria a um interessado durante a atividade de concepção.
- 5.7.** Desenvolva um "kit" de levantamento de requisitos. O kit deve incluir um conjunto de diretrizes para realizar uma reunião para levantamento de requisitos e materiais que podem ser utilizados para facilitar a criação de listas e quaisquer outros itens que poderiam ajudar na definição dos requisitos.
- 5.8.** Seu professor irá dividir a classe em grupos de quatro a seis alunos. Metade do grupo irá desempenhar o papel do departamento de marketing e a outra fará o papel da engenharia de software. Sua tarefa é definir os requisitos para a função de segurança do *CasaSegura* descrita neste capítulo. Realize uma reunião para levantamento de requisitos usando as diretrizes apresentadas neste capítulo.
- 5.9.** Desenvolva um caso de uso completo para uma das atividades a seguir:
- Fazer um saque em um caixa eletrônico.
 - Usar seu cartão de débito para uma refeição em um restaurante.
 - Comprar ações usando uma conta de corretagem on-line.
 - Procurar livros (sobre um assunto específico) usando uma livraria on-line.
 - Uma atividade especificada pelo seu professor.
- 5.10.** O que representam as "exceções" nos casos de uso?
- 5.11.** Descreva o que é um *padrão de análise* com suas próprias palavras.
- 5.12.** Usando o modelo apresentado na Seção 5.5.2, sugira um ou mais padrões de análise para os seguintes campos de aplicação:
- Software contábil
 - Software de e-mail
 - Navegadores para a Internet
 - Software de processamento de texto
 - Software para criação de sites
 - Um campo de aplicação especificado pelo seu professor
- 5.13.** Qual o significado de *ganha-ganha* no contexto das negociações durante uma atividade de engenharia de requisitos?
- 5.14.** O que você acha que acontece quando uma validação de requisitos revela um erro? Quem será envolvido na correção do erro?

LEITURAS E FONTES DE INFORMAÇÃO COMPLEMENTARES

Pelo fato de ser crucial para o sucesso na criação de qualquer sistema complexo baseado em computadores, a engenharia de requisitos é discutida em uma ampla gama de livros. Hood e seus colegas (*Requirements Management*, Springer, 2007) discutem uma série de questões da engenharia de requisitos que abrangem tanto sistemas quanto a engenharia de software. Young (*The Requirements Engineering Handbook*, Artech House Publishers, 2007) apresenta uma discussão aprofundada das tarefas da engenharia de requisitos. Wiegers (*More About Software Requirements*, Microsoft Press, 2006) aborda várias técnicas práticas para o gerenciamento e o levantamento de requisitos. Hull e seus colegas (*Requirements Engineering*,

2. ed., Springer-Verlag, 2004), Bray (*An Introduction to Requirements Engineering*, Addison-Wesley, 2002), Arlow (*Requirements Engineering*, Addison-Wesley, 2001), Gilb (*Requirements Engineering*, Addison-Wesley, 2000), Graham (*Requirements Engineering and Rapid Development*, Addison-Wesley, 1999) e Sommerville e Kotonya (*Requirements Engineering: Processes and Techniques*, Wiley, 1998) são apenas alguns exemplos dos muitos livros dedicados ao assunto. Gottesdiener (*Requirements by Collaboration: Workshops for Defining Requirements*, Addison-Wesley, 2002) fornece úteis orientações para aqueles que precisam estabelecer um ambiente colaborativo com seus interessados em termos de levantamento de requisitos.

Lauesen (*Software Requirements: Styles and Techniques*, Addison-Wesley, 2002) traz uma abrangente pesquisa sobre notação e métodos de análise para levantamento de requisitos. Weigers (*Software Requirements*, Microsoft Press, 1999) e Leffingwell e seus colegas (*Managing Software Requirements: A Use Case Approach*, 2. ed., Addison-Wesley, 2003) apresentam um útil conjunto das melhores práticas para levantamento de requisitos e sugerem diretrizes pragmáticas para a maioria dos aspectos do processo da engenharia de requisitos.

Uma visão da engenharia de requisitos baseada em padrões é descrita por Withall (*Software Requirement Patterns*, Microsoft Press, 2007). Ploesch (*Assertions, Scenarios and Prototypes*, Springer-Verlag, 2003) discute técnicas avançadas para desenvolvimento de requisitos de software. Windle e Abreo (*Software Requirements Using the Unified Process*, Prentice-Hall, 2002) discutem a engenharia de requisitos no contexto do Processo Unificado e da notação da UML. Alexander e Steven (*Writing Better Requirements*, Addison-Wesley, 2002) fornecem um breve conjunto de diretrizes para redação clara dos requisitos, representando-as como cenários e revisando o resultado final.

A modelagem de casos de uso muitas vezes é o motor para a criação de todos os demais aspectos do modelo de análise. O assunto é debatido exaustivamente por Rosenberg e Stephens (*Use Case Driven Object Modeling with UML: Theory and Practice*, Apress, 2007). Denny (*Succeeding with Use Cases: Working Smart to Deliver Quality*, Addison-Wesley, 2005), Alexander e Maiden (eds.) (*Scenarios, Stories, Use Cases: Through the Systems Development Life-Cycle*, Wiley, 2004), Leffingwell e seus colegas (*Managing Software Requirements: A Use Case Approach*, 2. ed., Addison-Wesley, 2003) apresentam um proveitoso conjunto das melhores práticas para levantamento de requisitos. Bittner e Spence (*Use Case Modeling*, Addison-Wesley, 2002), Cockburn [Coc01], Armour e Miller (*Advanced Use Case Modeling: Software Sistemas*, Addison-Wesley, 2000) e Kulak e seus colegas (*Use Cases: Requirements in Context*, Addison-Wesley, 2000) abordam o levantamento de requisitos com ênfase na modelagem de casos de uso.

Uma ampla gama de fontes de informação sobre análise e engenharia de requisitos se encontra à disposição na Internet. Uma lista atualizada de referências na Web, relevantes para a análise e engenharia de requisitos, pode ser encontrada no site www.mhhe.com/engcs/compsci/pressman/professional/olc/ser.htm.