Vinícius Souza

Engenharia de requisitos de software

COLEÇÃO EAD
Editora Unisinos

ENGENHARIA DE REQUISITOS DE SOFTWARE

VINÍCIUS COSTA DE SOUZA

EDITORA UNISINOS
2011

APRESENTAÇÃO

Um dos problemas mais críticos e recorrentes nos projetos de desenvolvimento de software está relacionado a erros ou falta da especificação dos requisitos do sistema. Como resultado desses problemas, temos requisitos que não atendem adequadamente às necessidades dos clientes, e algumas das razões para este insucesso são: falta de processos de desenvolvimento bem definidos; requisitos de software não bem compreendidos e acordados; uso de técnicas inadequadas; falta de uma ferramenta de apoio e a própria complexidade dos softwares a serem desenvolvidos.

Em função disso, muitas alterações em requisitos já acordados precisam ser realizadas, além da necessidade de inclusão de novos requisitos durante o processo de desenvolvimento, o que provoca retrabalho e impacta diretamente no prazo e no custo do projeto e reduz a qualidade do produto final.

Assim, esse livro apresenta uma visão geral do processo de desenvolvimento de software e os principais conceitos relacionados à engenharia de requisitos. Para isso, o livro está organizado em nove capítulos. O primeiro capítulo apresenta uma breve contextualização dos requisitos em um projeto de software, bem como as dificuldades e os desafios relacionados. Já no capítulo dois é apresentado o processo de software como um todo, através das suas atividades fundamentais e de apoio, bem como alguns modelos de processo e modelos de melhoria. O capítulo três apresenta o conceito de requisito de software, bem como sua possível classificação. No capítulo quatro são apresentas as atividades do processo de engenharia de requisitos, bem como algumas ferramentas de apoio disponíveis. Os capítulos cinco, seis, sete, oito e nove descrevem, respectivamente, as cinco atividades da engenharia de requisitos: elicitação, análise/ negociação, documentação, validação e gerenciamento de requisitos.

SUMÁRIO

CAPÍTULO 1

INTRODUÇÃO

Neste capítulo, você vai compreender qual a importância dos requisitos para os projetos de software, bem como conhecer as dificuldades e os desafios relacionados a essa importante etapa.

1.1 O papel dos requisitos no desenvolvimento de software

Os requisitos, em um projeto de software, são as funcionalidades que o sistema deverá possuir e as restrições sob as quais deverá operar. É a partir da documentação dos requisitos, que detalha como deve ser cada funcionalidade do sistema, bem como cada uma das caraterísticas e restrições desejadas, que se define o escopo do projeto. A esse documento, com a descrição detalhada dos requisitos chamamos de especificação de requisitos do sistema. Já a engenharia de requisitos, trata-se do processo que envolve todas as atividades exigidas para criar e manter a especificação de requisitos (KOTONYA e SOMMERVILLE, 1998).

A elaboração adequada da especificação contribui, significativamente, para a qualidade final de um projeto de software, visto que é a partir dela que as estimativas de equipe, prazo e custo são realizadas e o sistema é projetado, construído e testado. Um projeto que inicia com requisitos incompletos e incorretos terá que passar por vários replanejamentos para que tenha chance de ser concluído com sucesso e, consequentemente, terá seus custos e prazos aumentados em função do retrabalho necessário.

Assim, todos os requisitos identificados pelo analista de sistemas e acordados com o cliente devem ser documentados de forma clara e detalhada, visto que essa documentação é utilizada em todas as fases do desenvolvimento de um sistema e, até mesmo, após a sua entrega.

Os projetistas de software utilizam a especificação de requisitos para projetar a melhor estrutura para a construção de um sistema que atenda aos requisitos. Os desenvolvedores utilizam a documentação de requisitos, juntamente com o projeto do

sistema, para saber exatamente como e o que programar. Os testadores utilizam a especificação para planejar os testes e para verificar se o sistema foi construído exatamente conforme o especificado.

Entretanto, mesmo que a especificação de requisitos seja realizada de forma adequada, é natural que ao longo do ciclo de vida do software sejam necessárias alterações nos requisitos do sistema em função da necessidade de manutenções corretivas e evolutivas. Por isso, é extremamente importante que todos os requisitos tenham sido adequadamente documentados e mantidos. É baseando-se na especificação dos requisitos que o profissional ou equipe que realizará a manutenção poderá minimizar os riscos de causar impacto não desejado em outras partes do sistema.

Além disso, durante o desenvolvimento de softwares de grande porte e alta complexidade, inevitavelmente, teremos alterações nos requisitos. Nestes casos, podemos considerar que os requisitos são necessariamente incompletos durante o processo de desenvolvimento. O problema, inicialmente levantando, amadurece durante o desenvolvimento e acaba sendo compreendido com mais clareza, o que ocasiona necessidade de alterações nos requisitos.

Assim, tão importante quanto a adequada especificação dos requisitos é a realização constante de um eficaz processo de gerenciamento dos requisitos. O gerenciamento de requisitos visa compreender e controlar as mudanças nos requisitos de software, garantindo um melhor controle sobre estas mudanças e minimizando os riscos de impactos não desejados e não controlados.

1.2 Dificuldades e desafios

Entre as principais dificuldades para que uma correta especificação de requisitos seja realizada, podemos destacar:

- baixa compreensão do negócio do cliente;
- mudanças no negócio durante o desenvolvimento;
- falta de envolvimento dos usuários;
- dificuldades de comunicação com os usuários;
- mudanças no pessoal envolvido durante o processo;
- complexidade do sistema;
- não utilização de técnicas adequadas;

- falta de ferramentas de apoio;
- falta de gerenciamento de requisitos.

Para que um sistema possa ser especificado com qualidade, é extremamente necessário que o analista de sistemas compreenda o negócio do cliente, para o qual o sistema dará apoio, visto que, somente assim, o analista poderá elaborar a solução mais adequada para as necessidades do cliente.

A dificuldade, neste caso, refere-se à impossibilidade de os analistas de sistemas compreenderem os mais diversos tipos de negócios, como, por exemplo, imóveis, varejo, saúde, marketing, recursos humanos, finanças, entre tantos outros. Por isso, em muitos projetos, os analistas dependem do auxílio dos clientes ou, algumas vezes, de consultores especializados em determinada área de negócio, os chamados analistas de negócios.

Além disso, em muitos projetos ocorrem mudanças no negócio do cliente durante o desenvolvimento do sistema e, nestes casos, o analista deve estar atento para perceber e compreender essas mudanças, realizando as alterações necessárias nos requisitos.

Contudo, em muitos projetos os usuários não têm tempo disponível ou interesse em contribuir com o analista de sistemas, visto que estão preocupados em resolver os problemas atuais do seu trabalho ou apresentam resistência ao projeto, por sentirem-se ameaçados pelo novo sistema. Também ocorre, algumas vezes, que a pessoa que está contribuindo significativamente para o projeto se desliga da empresa, o que prejudica consideravelmente o projeto.

Um outro problema, muito comum e de alta gravidade, na etapa de especificação de requisitos, é a dificuldade de comunicação entre os analistas, clientes e a equipe do projeto. Em muitos casos, os usuários não conseguem se expressar de forma clara, o que dificulta muito a compreensão do analista sobre o negócio e sobre as suas necessidades. Além disso, o analista precisa ser muito claro e preciso na documentação dos requisitos para que, tanto o cliente como a equipe de desenvolvimento possam compreender com clareza os requisitos.

Os problemas relacionados à falta de técnicas e ferramentas adequadas, por parte do analista, também causam prejuízos para o projeto. Isso ocorre, visto que requisitos deixam de ser identificados ou são inadequadamente controlados, causando retrabalho e, consequentemente, atrasos e aumento dos custos do projeto.

Somando isso aos fatores já citados, devemos considerar a complexidade do próprio sistema a ser especificado. Atualmente, a grande maioria das organizações e

atividades humanas são fortemente baseadas em software e, com isso, a complexidade dos projetos tem aumentado consideravelmente, sendo bastante comum projetos com duração superior a doze meses, envolvendo equipes com mais de cinquenta pessoas.

CAPÍTULO 2

REQUISITOS DE SOFTWARE

Este capítulo apresenta o conceito de requisito de software, bem como sua classificação em duas categorias: funcionais e não funcionais. Além disso, os requisitos de software também podem ser classificados como estáveis e voláteis. No caso dos requisitos não funcionais, existem subcategorias que os classificam em requisitos não funcionais de produto, processo e externos.

Os requisitos do software são as descrições dos principais recursos de um produto de software, ou seja, as funções e restrições do sistema, e fornecem uma estrutura básica para o desenvolvimento de um produto de software (PRESSMAN, 2001). Tratam-se das funcionalidades que o sistema deve possuir e as restrições sob as quais deve operar (KOTONYA e SOOMERVILLE, 1998).

O requisito é o principal item da engenharia de requisitos. Ele deve ser escrito de forma precisa e clara, para que seja possível entender exatamente o que deve ser construído. Assim, os requisitos descrevem como deve ser cada função do sistema e não como deve ser sua implementação, pois isso será definido na fase de projeto e implementação do software.

É muito importante que os requisitos sejam entendíveis não somente por parte da organização, mas também por parte do cliente, para que cumpram efetivamente aquilo que o cliente deseja. Para isso, é importante que o documento de requisitos do projeto seja claro e organizado, o que é possível através da classificação dos requisitos.

Diferentes autores propõem diferentes formas de classificar os requisitos de um projeto de software. Sommerville (2005) classifica os requisitos da seguinte forma:

- *Requisitos do usuário*: declarações, em linguagem natural e diagramas, sobre os serviços que o sistema oferece e as restrições para a sua operação. Escrito para os clientes;
- *Requisitos do sistema*: estabelecem detalhadamente as funções e

restrições do sistema. O documento de requisitos, chamado de especificação funcional, pode servir como um contrato entre cliente e desenvolvedor.

Além disso, tanto os requisitos de usuário como os de sistema são subdivididos em requisitos funcionais e não funcionais e podem, também, ser classificados como requisitos voláteis ou estáveis.

2.1 Requisitos funcionais

Os requisitos funcionais (RF) são descrições das funcionalidades ou serviços que se espera que o sistema forneça. Também podem ser documentados os serviços que não serão disponibilizados pelo sistema. Dependem do tipo de software que será desenvolvido, das solicitações dos usuários e das necessidades do negócio.

Alguns exemplos de requisitos funcionais são:

- cadastro de clientes;
- relatório de vendas;
- pesquisa de produtos;
- escolha de produtos;
- seleção da forma de pagamento.

2.2 Requisitos não funcionais

Como já mencionando anteriormente, os requisitos não funcionais (RNF) tratam-se da descrição das restrições sobre as quais o sistema deve operar. São as restrições sobre os serviços ou as funções oferecidas pelo sistema.

Diferentemente dos funcionais, esses requisitos não dizem respeito diretamente às funções do sistema, mas sim sobre as suas restrições. Muitos RNF dizem respeito ao sistema como um todo e não a características individuais do sistema, e por isso podem ser até mais importantes que os RF.

Os RNF devem ser objetivos e não podem expressar desejo ou opinião pessoal. Isso porque esses requisitos devem ser validados e, para isso, precisam ser definidos de forma que possam ser testados.

Definir que o sistema deve ser fácil de usar é um exemplo incorreto de requisito

não funcional, visto que a facilidade de uso de um sistema é algo muito subjetivo e pessoal. O correto, neste caso, é definir que qualquer serviço do sistema seja realizado em até quatro etapas ou que o tempo necessário para treinar novos usuários do sistema não seja superior a duas horas. Dessa forma, é possível validar, ao final do projeto, se o sistema cumpre ou não esses requisitos.

Para facilitar a organização dos requisitos não funcionais, existem diferentes propostas de classificação. Cabe a cada empresa definir as categorias que deseja utilizar a fim de padronizar a especificação de requisitos e tornar seus projetos mais organizados.

Sommerville (2005) sugere classificar os requisitos em três categorias iniciais: requisitos de produto, requisitos de processo e requisitos externos, as quais se subdividem em outras categorias, conforme ilustrado na figura 1.

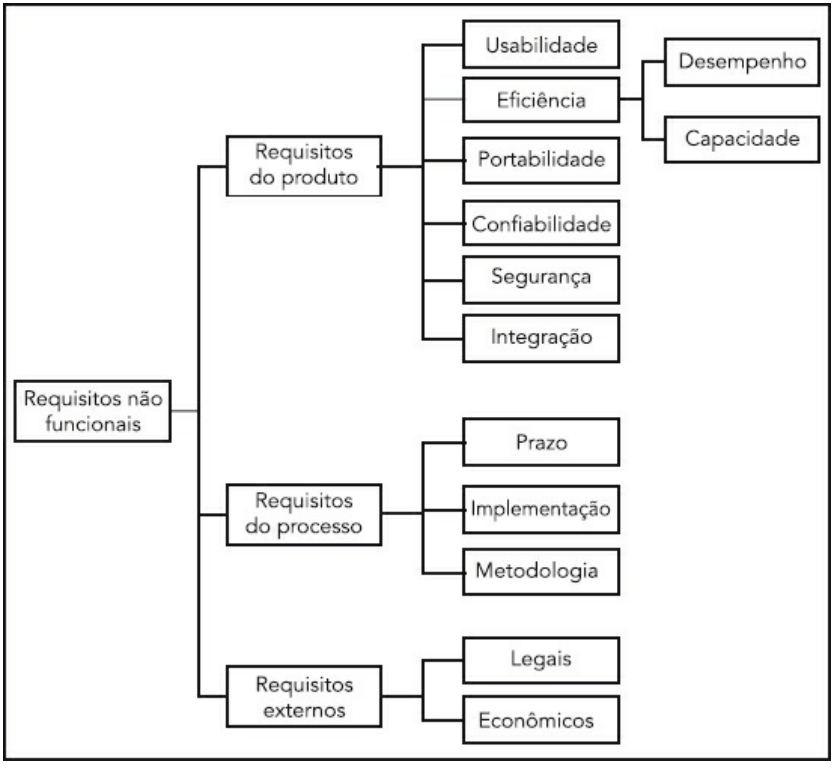

Figura 1 – Classificação dos requisitos não funcionais.
Fonte: adaptado de Sommerville, 2005.

Os requisitos de produto especificam características do comportamento do produto como, por exemplo, em quais sistemas operacionais o software poderá executar, qual é o tempo máximo de resposta para as operações do sistema, entre outras.

O quadro 1 apresenta um exemplo de requisito não funcional para cada um das categorias de requisitos do produto.

Quadro 1 – Exemplos de requisitos não funcionais de produto

Categoria	Exemplo
Usabilidade	Todas as funcionalidades do sistema devem oferecer uma opção de ajuda ao usuário.
Desempenho	O tempo máximo de resposta do sistema deve ser de 40 segundos.
Capacidade	O sistema deve suportar até dez mil usuários simultâneos.
Portabilidade	O sistema deverá ser executado nos sistemas operacionais Windows e Linux.
Confiabilidade	O sistema deve se recuperar de falhas de forma integra, ou seja, não podem ser gravados dados incompletos no sistema.
Segurança	O sistema deve criptografar os dados transferidos via rede externa
Integração	O sistema a ser desenvolvido deve ser integrado como atual sistema financeiro do cliente.

Já os requisitos não funcionais de processo são provenientes das políticas e dos padrões organizacionais, tanto da empresa do cliente como da desenvolvedora do software. O quadro 2 apresenta exemplos de requisitos do processo.

Quadro 2 – Exemplos de requisitos não funcionais de processo

Categoria	Exemplo
Prazo	O sistema deve ser entregue em até oito meses.
Implementação	O sistema deve ser desenvolvido em PHP.
Metodologia	O sistema deve ser desenvolvido no paradigma orientado a objetos.

Os requisitos não funcionais externos são aqueles definidos por entidades

diferentes do cliente e da equipe de desenvolvimento, geralmente associados a questões legais ou econômicas. Nesse caso, as alterações nos requisitos independem da vontade do cliente ou do fornecedor do software. O quadro 3 apresenta exemplos de requisitos não funcionais externos.

Quadro 3 – Exemplos de requisitos não funcionais externos

Categoria	Exemplo
Legais	O sistema deve estar de acordo com as atuais alíquotas do Imposto de Renda brasileiro.
Econômicos	O sistema deve utilizar a taxa de crescimento da Bolsa de Valores de São Paulo.

É importante destacar que nem sempre é possível fazer uma distinção clara entre os RF e os RNF. Um mesmo requisito pode ser classificado como RF ou RNF, dependendo do nível de detalhamento do mesmo.

Se definirmos que o sistema deve garantir que os dados sejam acessados apenas por pessoas autorizadas, estamos definindo um RNF, pois trata-se de uma característica do sistema como um todo e não de uma funcionalidade. Entretanto, se definirmos que o sistema deve possuir uma funcionalidade para autenticação na qual os usuários devem se identificar através de um nome de usuário e senha, estamos definindo uma função a ser disponibilizada pelo sistema e, neste caso, um RF.

Assim, é comum requisitos não funcionais mais abstratos serem decompostos em requisitos funcionais mais detalhados. Além disso, para atender a um RNF podem ser necessários vários RF.

2.3 Requisitos estáveis e voláteis

Mudanças nos requisitos ocorrem enquanto eles estão sendo elicitados, analisados, validados e depois do sistema entrar em produção. Essas mudanças são inevitáveis, o que não significa que o processo de engenharia de requisitos adotado tenha sido falho. Elas resultam da combinação de uma série de fatores como evolução do entendimento dos analisas, mudanças nas prioridades do cliente, mudanças organizacionais, tomada de decisão tardia, entre outras.

Embora mudanças sejam inevitáveis, são elas que fazem com que alguns requisitos sejam mais estáveis que outros. Requisitos estáveis estão relacionados com a essência do sistema e seu domínio de aplicação. Eles mudam mais moderadamente

que requisitos voláteis, os quais são específicos da concepção de um sistema em um ambiente particular e para um cliente particular.

De acordo com Kotonya e Sommerville (1998), existem pelo menos quatro tipos de requisitos voláteis: mutáveis, emergentes, supostos e dependentes.

Os requisitos *mutáveis* são aqueles que mudam devido às mudanças no ambiente em que operam, como os requisitos não funcionais externos relacionados a taxas e impostos.

Já os requisitos *emergentes* são aqueles que não podem ser completamente definidos enquanto o sistema está sendo especificado.

Os requisitos *supostos* são aqueles, declaradamente, baseados em suposições do usuário sobre como o sistema deve operar.

Finalmente, os requisitos *dependentes* são aqueles que dependem da definição de algum equipamento ou processo.

Assim, uma boa prática da engenharia de requisitos é a identificação dos requisitos que provavelmente sofrerão mudanças. Isso envolve classificar os requisitos, para identificar os mais voláteis e então analisar possíveis mudanças.

PARA COMPLEMENTAÇÃO DE ESTUDO

Utilize as referencias Kotonya e Sommerville (1998), Sommerville (2005) e Pressman (2001) para complementar seu estudo sobre requisitos de software.

CAPÍTULO 3

PROCESSO DE SOFTWARE

Este capítulo apresenta o conceito de processo de software, bem como suas metas e as condições para que elas sejam alcançadas com sucesso. Além disso, são apresentadas suas atividades fundamentais e de apoio.

3.1 Conceito, objetivos e condições

O processo de software é definido como um conjunto de atividades, métodos, práticas e transformações que pessoas empregam para desenvolver e manter software e produtos associados (HUMPHREY, 2004).

Segundo Pressman (2001), o processo de software envolve a definição dos métodos, ferramentas e atividades necessárias para o desenvolvimento e a manutenção de software, sendo os métodos definidos como a forma com que o processo é executado, as ferramentas sendo aquelas que oferecem apoio automatizado ou semiautomatizado aos métodos, aos processos e às atividades necessárias para que o software seja construído e mantido.

Já Sommerville (2005) define processo de software como um conjunto de atividades e resultados associados que levam à produção de um produto de software. Esse processo pode envolver o desenvolvimento de software desde o início, embora, cada vez mais, ocorra o caso de um software novo ser desenvolvido mediante a expansão e a modificação de sistemas já existentes.

Quanto aos objetivos, pode-se definir que todo o projeto de software deve entregar um sistema que atenda as necessidades e expectativas dos usuários, cumprindo as seguintes metas, definidas previamente:

- prazo;
- custo;
- qualidade.

Contudo, sabe-se que os projetos de software são extremamente complexos e envolvem muitas variáveis que, se não devidamente controladas, acarretam o insucesso do projeto.

Assim, após inúmeros projetos malsucedidos, com resultados inadequados, tanto em relação a custo e prazo como em relação à qualidade final, muitas empresas perceberam a importância de definir seus processos e utilizar métodos e ferramentas adequadas.

Por isso, conforme ilustrado na figura 2, as três condições fundamentais para que um projeto de software obtenha sucesso são: estabelecer uma equipe com pessoas capacitadas e motivadas, possuir processos de trabalho definidos e seguidos por todos e disponibilizar ferramentas de apoio adequadas.

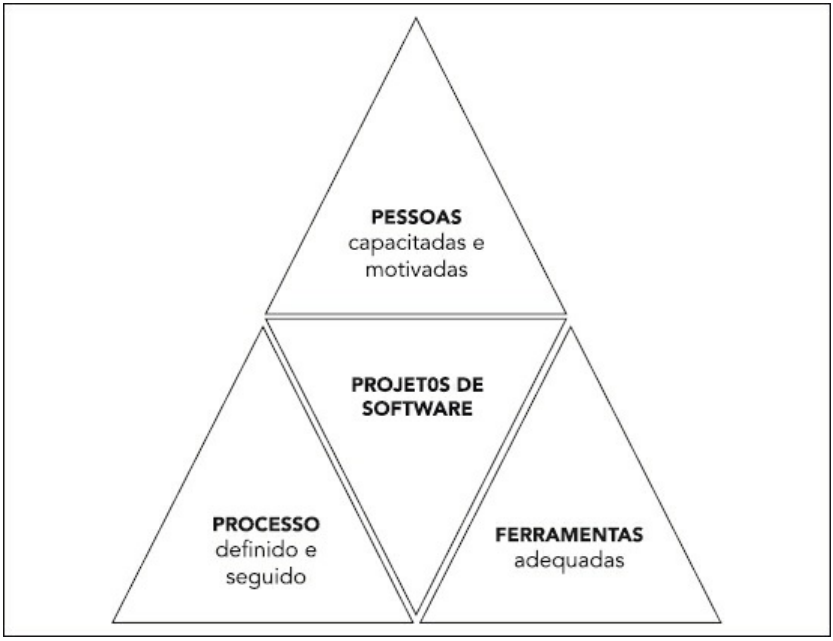

Figura 2 – Condições para projetos de software bem-sucedidos.
Fonte: Autor.

Além de possuir uma equipe com a quantidade de pessoas necessária, é preciso

que todos os membros da equipe estejam motivados e capacitados para realizar o trabalho, tanto do aspecto técnico como processual. Para isso, deve-se garantir que todos já possuam o conhecimento técnico e sobre o processo de trabalho ou que sejam treinados previamente.

Com relação à motivação, deve-se oferecer condições de trabalho e benefícios suficientes para que as pessoas estejam satisfeitas com seu atual emprego, minimizando as chances de resultados de baixa qualidade e a perda de membros da equipe durante a execução do projeto.

Todavia, apenas pessoas capacitadas e motivadas não são suficientes para que um projeto de software obtenha sucesso. É preciso que existam processos de trabalho definidos, os quais facilitam e organizam o trabalho da equipe, e que esses processos sejam de fato seguidos por todos. Como já mencionado no Capítulo 4, para que um processo seja seguido se faz necessária à realização de auditorias periódicas, as quais garantem que o processo definido esteja realmente sendo executado por todos.

Para que o projeto tenha produtividade e se atinja a qualidade esperada, são necessárias ferramentas de apoio adequadas. Atualmente, existem muitas ferramentas de apoio ao processo de software, disponíveis tanto comercialmente como gratuitas. Assim, o desafio para as empresas é definir quais são as ferramentas melhor adequadas para a suas características e necessidades.

Contudo, não há um processo de software ideal e, assim, diferentes empresas definem abordagens inteiramente diferentes para o desenvolvimento de software. Em todas as empresas os processos evoluem para explorar a capacidade das pessoas, assim como em função do tipo de sistema que é desenvolvido.

Por isso, diferentes autores classificam as atividades do processo de software de diversas formas. Para Sommerville (2005), embora existam muitos processos de software diferentes, há atividades fundamentais comuns a todos eles. São elas: *Especificação*, na qual são definidas as suas funcionalidades e restrições; *Projeto e Implementação*, na qual o software é produzido de acordo com a especificação; *Validação*, na qual o software é validado para garantir que faz o que o cliente deseja, e a *Evolução*, na qual o software é adaptado para atender às necessidades mutáveis do cliente.

Nesse livro, as atividades do processo de software foram classificadas em fundamentais e de apoio. As atividades fundamentais representam as atividades técnicas necessárias para a construção e manutenção de software, sendo elas análise, projeto, implementação, testes, implantação e manutenção. Já as atividades de apoio são atividades necessárias para garantir que as atividades fundamentais sejam

realizadas da melhor forma possível, sendo elas a gerência de projetos, a gerência da configuração e a gerência da qualidade.

3.2 Atividades fundamentais

Como já mencionado anteriormente, as atividades fundamentais do processo de software são as atividades técnicas necessárias para que o software seja construído e mantido.

3.2.1 Análise

A análise, também chamada de análise de sistemas ou especificação de sistemas, a qual é o foco deste livro, trata-se da primeira e fundamental atividade do processo de software. É através desta atividade que o analista de sistemas, em conjunto com o cliente e os usuários, define as funcionalidades e restrições do sistema que será desenvolvido, o que chamamos de requisitos do sistema.

Para isso, o analista deve obter o maior número possível de informações, como os objetivos de negócio do cliente, as necessidades dos usuários, os sistemas já existentes na empresa, as regras e os regulamentos da área, além de informações sobre a área de negócio para a qual o sistema será desenvolvido. A figura 3 ilustra o processo de análise de um sistema, apresentando suas entradas e saídas.

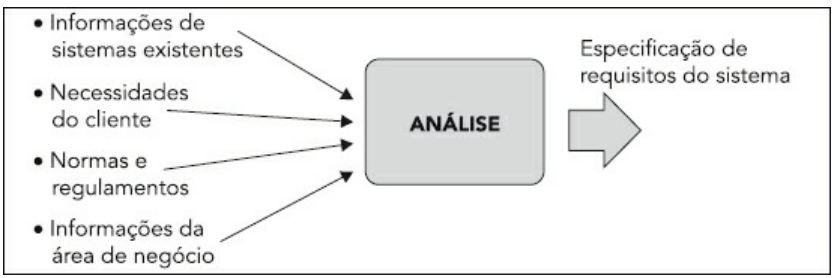

Figura 3 – Processo de análise de um sistema.
Fonte: adaptado de Kotonya e Sommerville, 1998.

Quanto maior o número de requisitos identificados e especificados corretamente

nessa etapa, melhores são as estimativas de prazo e custo e maiores são as chances do software ser desenvolvido com qualidade e atender às necessidades e expectativas do cliente.

Entretanto, isso não impede que muitos requisitos sejam identificados durante a construção do sistema, o que de certa forma facilita a identificação de novas funcionalidades e restrições, visto que tanto o cliente como a equipe de desenvolvimento obtêm uma melhor compreensão do sistema que está sendo desenvolvido.

Como resultado da atividade de análise, temos um documento com a descrição detalhada, em linguagem de usuário, de cada uma das funcionalidades e restrições do sistema. Esse documento é comumente chamado de Especificação de Requisitos do Sistema.

A análise de um sistema subdivide-se em cinco atividades específicas, sendo elas elicitação, análise/negociação, documentação, validação e gerenciamento de requisitos, as quais são detalhadas nos próximos capítulos deste livro.

3.2.2 Projeto

A etapa de projeto de um sistema, também chamada de engenharia ou arquitetura de software, é a etapa na qual o projetista, engenheiro ou arquiteto do sistema projeta uma estrutura de software que atenda da melhor forma possível à especificação de requisitos.

Um projeto de software é uma descrição da estrutura de software a ser construída, da estrutura dos dados a serem armazenados e processados, das interfaces entre os componentes do sistema e, algumas vezes, dos algoritmos a serem utilizados.

A etapa de projeto pode envolver o desenvolvimento de vários modelos do sistema, em diferentes níveis de abstração, podendo ser uma especificação abstrata, produzida para esclarecer os requisitos, ou uma especificação mais detalhada e precisa de como o sistema deve ser estruturado. Assim, alguns exemplos de itens projetados nesta etapa são:

- *Arquitetura do sistema*: os subsistemas que constituem o sistema e suas relações são definidos e documentados;
- *Subsistemas*: para cada subsistema é projetada uma estrutura que contemple suas funções e restrições;
- *Interfaces*: para cada subsistema, sua interface com outros

subsistemas é projetada e documentada;

- *Componentes*: são projetados os componentes do sistema que contemplarão as diversas funções especificadas nos requisitos. Cada componente possuirá um conjunto de funções do sistema;
- *Estrutura de dados*: a estrutura para o armazenamento dos dados do sistema é detalhadamente projetada e documentada;
- *Algoritmos*: os algoritmos para os serviços mais complexos são projetados;

Apesar disso, é muito comum, nas organizações, que o desenvolvimento dos sistemas inicie a partir de um conjunto prévio de requisitos, para o qual um projeto informal é preparado. Após, esse projeto é modificado e complementado à medida que o sistema é desenvolvido e novos requisitos são definidos. Neste caso, se não houver um adequado controle dessas mudanças, ao final, o projeto documentado é uma descrição incoerente e incompleta do sistema.

Uma abordagem mais metódica para a etapa de projeto de sistemas é proposta pelos métodos estruturados, que são conjuntos de notações e diretrizes para o projeto de software. Alguns exemplos de métodos estruturados para o projeto de sistemas são: Projeto Estruturado (CONSTANTINE e YORDOUN, 1979), Análise de Sistemas Estruturados (GANE e SARSON, 1979) e Desenvolvimento de Sistemas (JACKSON, 1983). Além desses, um dos métodos mais utilizados, atualmente, é o do projeto orientado a objetos (ROBINSON, 1992; BOOCH, 1994; RUMBAUGH *et al.*, 1991; BOOCH *et al.*,1999).

Embora esse livro não tenha como objetivo apresentar os métodos de projeto de software, a seguir são ilustrados exemplos dos possíveis diagramas elaborados em projetos orientados a objetos, através da notação UML.

A figura 4 apresenta o diagrama de casos de uso, que tem como objetivo auxiliar a comunicação entre os analistas e o cliente. Esse diagrama apresenta as funcionalidades do sistema, do ponto de vista do cliente.

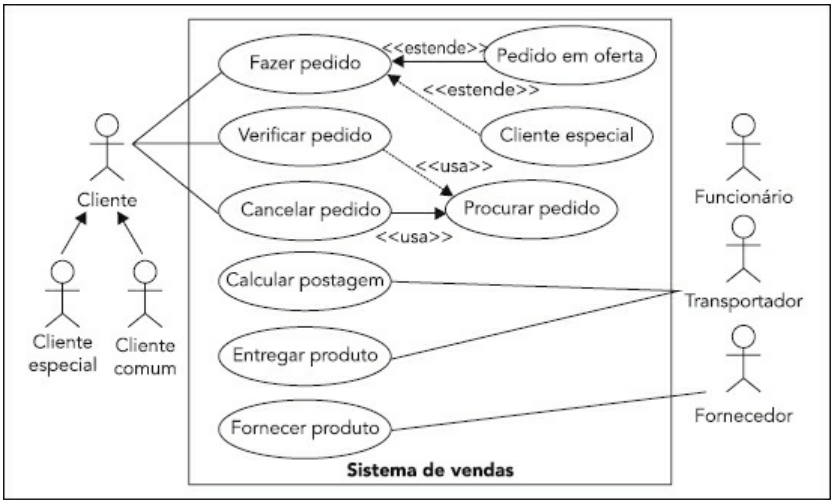

Figura 4 – Exemplo de um diagrama de casos de uso.
Fonte: Autor.

Um dos diagramas mais utilizados para projetar software orientado a objetos é o diagrama de classes, ilustrado na figura 5, que apresenta os diversos tipos de objetos do sistema e o relacionamento entre eles.

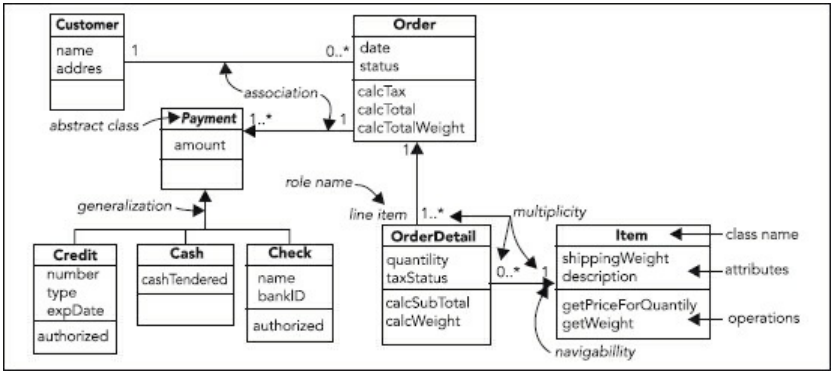

Figura 5 – Diagrama de classes.

Fonte: www.distancelearnig101.net/interm_prog/uml/.

A figura 6 apresenta o diagrama de sequência, o qual tem como objetivo mostrar como as mensagens entre os objetos são trocadas, no decorrer do tempo, para a realização de uma operação.

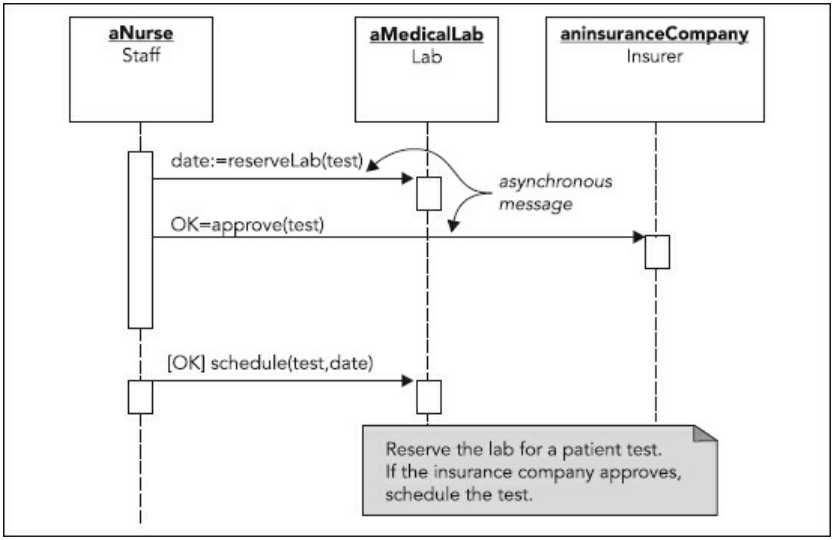

Figura 6 – Exemplo de diagrama de sequência.
Fonte: www.distancelearnig101.net/interm_prog/uml/.

Já a figura 7 ilustra um diagrama de colaboração, o qual tem o mesmo objetivo do diagrama de sequência, porém neste o tempo não é mais representado por linhas verticais, mas sim através de uma numeração.

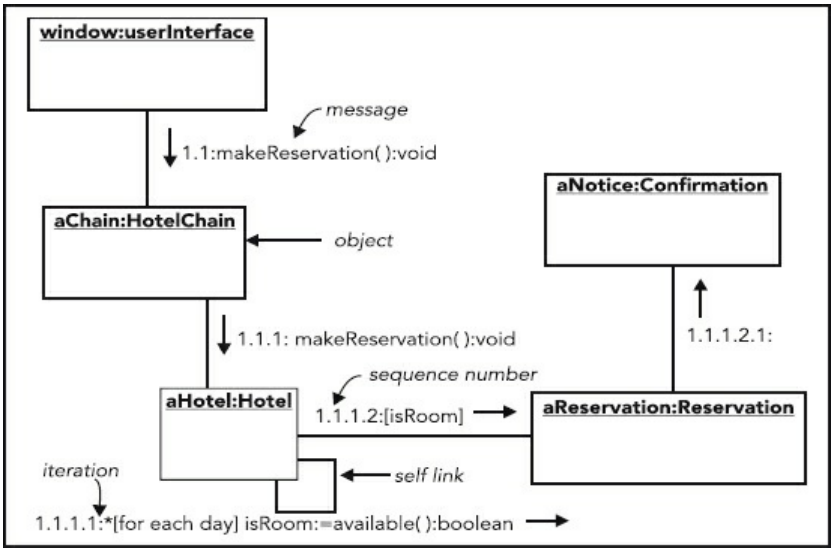

Figura 7 – Diagrama de colaboração.
Fonte: www.distancelearnig101.net/interm_prog/uml/.

O diagrama de estado, apresentado na figura 8, mostra os possíveis estados de um objeto e as transações responsáveis pelas suas mudanças de estado.

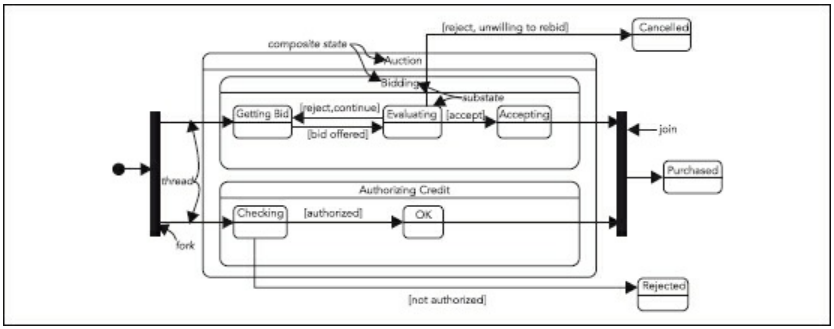

Figura 8 – Diagrama de estado.
Fonte: www.distancelearnig101.net/interm_prog/uml/.

O diagrama de atividade, apresentado na figura 9, tem como objetivo mostrar o fluxo de atividades em um processo.

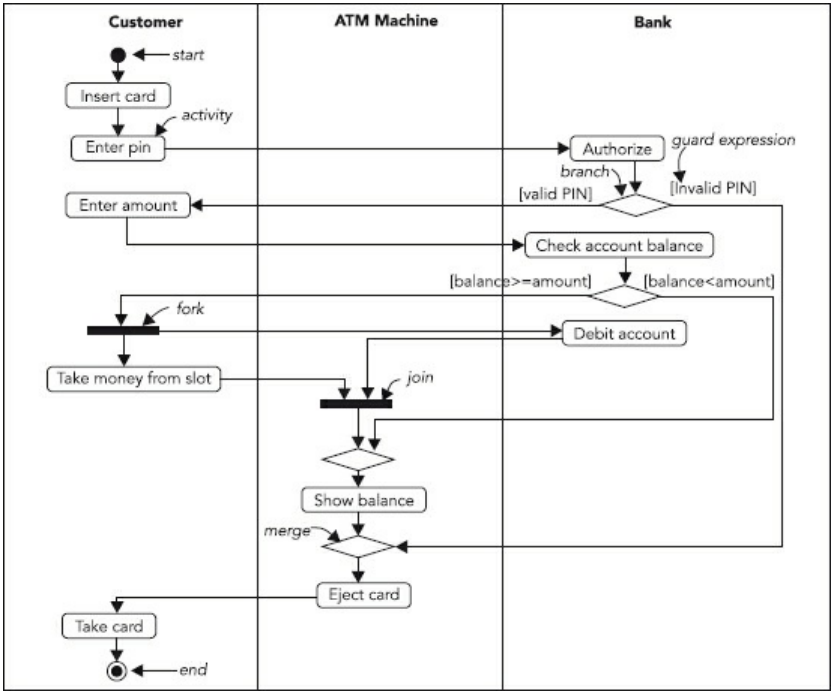

Figura 9 – Diagrama de atividades.
Fonte: www.distancelearnig101.net/interm_prog/uml/.

A figura 10 apresenta um exemplo do diagrama de componentes, o qual tem como objetivo definir o agrupamento das classes de um sistema em pacotes.

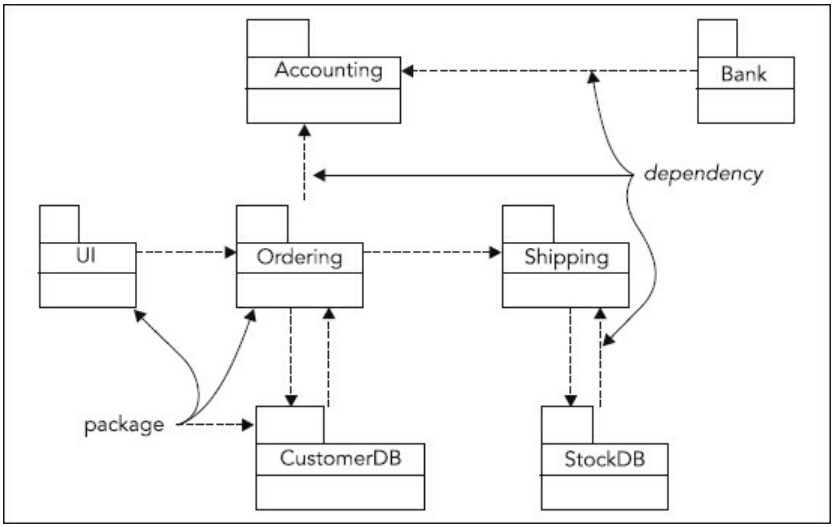

Figura 10 – Diagrama de componentes.
Fonte: www.distancelearnig101.net/interm_prog/uml/.

Finalmente, a figura 11 apresenta o diagrama de implantação que mostra a configuração de nós de processamento, em tempo de execução, e os componentes que neles existem.

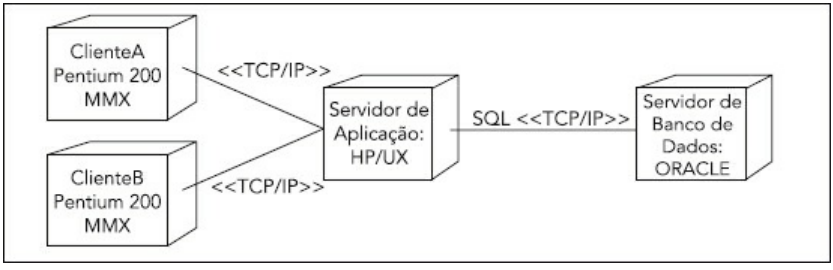

Figura 11 – Exemplo de diagrama de implantação.
Fonte: www.distancelearnig101.net/interm_prog/uml/.

3.2.3 Implementação

A implementação trata-se da etapa de conversão de um conjunto de especificações em sistema executável. É a etapa em que o sistema é de fato construído através da sua programação. A implementação é um processo, geralmente individual, que deve seguir determinados padrões para que produza resultados de fácil manutenibilidade.

Para a realização dessa atividade existem diferentes tecnologias disponíveis, como as linguagens de programação Java, C, PHP, C#, C++, Delphi, entre tantas outras.

Além das linguagens de programação, existem os diferentes paradigmas, ou seja, diferentes formas de estruturar e executar programas de computador. Alguns dos principais paradigmas de programação são apresentados a seguir.

- **Imperativo:** paradigma associado a comandos sequenciais e a troca de estados de memória. Alguns exemplos de linguagens que permitem a programação no paradigma imperativo são: Cobol, C, Pascal, PHP, entre outras.
- **Orientado a objetos:** associação de comportamento a estruturas de dados chamados objetos. Exemplos de linguagens que permitem a programação orientada a objetos são: C++, Java, PHP, entre outras.
- **Concorrente**: trata-se da programação multitarefa, ou seja, o programa é criado para que mais de uma tarefa seja disparada simultaneamente, as quais são executados de forma concorrente, alternando o uso do processador. Java, C++ e Ruby são exemplos de linguagens que permitem a programação concorrente.
- **Lógico**: neste paradigma a computação é o resultado de funções lógicas, baseadas em fatos, regras e questões. A principal linguagem para programação lógica é o Prolog.
- **Funcional:** neste caso, a computação é o resultado de funções matemáticas, representadas através de listas. A principal linguagem utilizada para a programação neste paradigma é o Lisp.

Além desses paradigmas, algumas tendências atuais, na área de programação, são a programação orientada a serviços, programação paralela, programação segura, programação orientada a aspectos e programação ágil, entre outras.

3.2.4 Testes

A etapa de testes trata-se da execução de um sistema com o objetivo de encontrar falhas. O quadro 4 apresenta algumas definições importantes de serem diferenciadas sobre testes de software.

Quadro 4 – Definições de termos sobre teste de software

Termo	Definição
Teste	Processo de execução de um sistema, ou parte dele, com o objetivo de encontrar falhas.
Verificação	Teste para encontrar falhas em um sistema, em ambiente de testes ou simulado.
Validação	Teste para encontrar falhas em um sistema, em ambiente real.
Debug	Processo para descobrir a causa de uma falha encontrada.

Alguns dos princípios fundamentais da área de teste de software são apresentados a seguir:

- um bom caso de teste deve possuir alta probabilidade de encontrar erros e não de provar que o sistema funciona;
- um sistema não deve ser testado por quem o desenvolveu, visto que esse é o caso de menor probabilidade de se encontrar erros;
- todo o caso de teste deve especificar o resultado esperado, a fim de permitir que o testador saiba se o teste obteve sucesso ou não;
- deve-se criar casos de teste tanto para entradas válidas como para as inválidas;
- como testar todas as possibilidades de uso do sistema é praticamente impossível, devemos testar os casos mais importantes;
- deve-se cometer erros, propositalmente, para verificar o comportamento do sistema.

Entre os principais tipos de teste de software podemos citar o teste unitário, teste funcional, teste integrado, teste de regressão e o teste de sistema.

Teste unitário

O teste unitário, também chamado de caixa branca (do inglês *White Box*), tem como objetivo examinar a estrutura interna do sistema e, por isso, exige que o testador

possua conhecimentos sobre a linguagem de programação utilizada na sua construção e sobre a estrutura interna do sistema.

O objetivo é garantir que cada caminho do programa seja executado pelo menos uma vez, a fim de garantir que não existem erros, os quais podem causar paradas no sistema.

Teste funcional

O teste funcional, também chamado de caixa preta (do inglês *Black Box*), é baseado nos requisitos do sistema e não exige que o testador possua conhecimentos sobre a linguagem de programação ou sobre a estrutura interna do sistema. Entretanto, esse tipo de teste exige uma boa documentação dos requisitos do sistema e habilidade para determinar o número de testes adequados. Trata-se do teste das funcionalidades do sistema, via interface de usuário, a fim de verificar se o sistema funciona e se está de acordo com a especificação dos requisitos.

Teste integrado

Trata-se do processo de testar se as várias partes de um sistema funcionam, de forma integrada, sem gerar erros. Testa-se as interfaces entre as partes do sistema (módulos, componentes, subsistemas etc.).

Teste regressivo

Tem como objetivo detectar problemas causados por mudanças ou adição de novas funcionalidades em módulos já testados. O desafio, nesse tipo de testes, é selecionar os casos de teste que devem ser reexecutados a cada alteração ou adição de uma nova funcionalidade.

Teste de sistema

Tratam-se dos testes realizados no sistema depois que o mesmo está pronto. Para isso, os objetivos do sistema devem estar extremamente claros, indicando exatamente o que os usuários esperam do sistema. Necessita da interação entre a equipe de testes e alguns usuários.

Alguns tipos de teste de sistema são o teste de carga, de configuração, compatibilidade, segurança, performance, instalação, confiabilidade, recuperação,

utilidade e usabilidade.

O quadro 5 apresenta um resumos dos principais tipos de teste de software.

Quadro 5 – Definições de termos sobre teste de software

Tipo de teste	Objetivo
Unitário	Testar a estrutura interna do sistema.
Funcional	Testar se o sistema funciona e se está em conformidades com a especificação.
Integrado	Testar se o sistema funciona após a junção de suas partes.
Regressivo	Testar se o sistema continua funcionando após mudanças ou adição de novas funcionalidades.
Sistema	Testar o sistema como um todo para verificar se está funcionando e de acordo com as necessidades dos usuários.

Antes de iniciarmos qualquer tipo de teste é necessário elaborar um plano de testes, no qual definimos as funcionalidades, regras e métodos a serem utilizados nos testes. Os principais elementos existentes em um plano de testes são: objetivos para cada fase de teste, prazos e responsáveis por cada atividade, padrões para a execução dos testes e registros dos resultados e os critérios para mensurar a realização e o sucesso dos testes.

É importante destacar que cada teste deve ser conduzido como um experimento, o qual deve ser cuidadosamente controlado e documentado, de tal forma que possa ser reproduzido.

As informações típicas de um relatório de resultado de testes são:

- nome do projeto;
- componentes e módulos testados;
- propósito dos testes;
- plano de testes;
- pessoas envolvidas;
- artifícios utilizados;
- configuração de hardware e software;
- resultados de cada teste, com os erros encontrados;
- assinatura do responsável pelos testes.

3.2.5 Implantação

Nesta etapa, todo o ambiente do cliente é preparado, o sistema é instalado e configurado e os usuários são treinados. Na implantação de um sistema, geralmente servidores de banco de dados e de aplicação são preparados e configurados para que o sistema possa ser utilizado. Além disso, são realizados os treinamentos que capacitam os usuários para a utilização do sistema. Essa etapa é encerrada somente quando tudo estiver pronto para que o sistema possa ser utilizado no cliente.

3.2.6 Manutenção

Trata-se de uma atividade que contempla a execução de todas as anteriormente descritas, para os casos em que um sistema já em uso tenha necessidade de alguma alteração, seja em função de uma correção ou de uma nova necessidade do cliente. As alterações para correção chamamos de manutenção corretiva e as alterações para inclusão de novos serviços de manutenção evolutiva.

Em ambos os casos, a alteração do sistema deve ser cuidadosamente conduzida, a fim de evitar que alterações em uma parte afetem, indevidamente, outras partes.

Cada alteração deve ser conduzida como um miniprojeto, para o qual deve-se especificar os requisitos, adaptar o projeto do sistema, implementar e testar a alteração e as demais partes do sistema. Após os testes, deve-se gerar uma nova versão do sistema e implantar no cliente.

3.3 Atividades de apoio

3.3.1 Gerência de projetos

A gerência de projetos tem como objetivo principal realizar o planejamento e o controle de todas as atividades necessárias para a realização de um projeto, seja ele de software ou de outra área qualquer.

Projeto é a denominação dada a um empreendimento temporário com a finalidade de produzir um produto ou serviço único. O fato de um projeto ser temporário não implica ter curta duração, mas sim a existência de início e fim predeterminados (PMI, 2004). A gerência de projetos consiste na aplicação de conhecimentos, habilidades, ferramentas e técnicas às atividades do projeto, de forma a atingir as necessidades e

expectativas das partes envolvidas.

Podemos definir um projeto como bem-sucedido quando ele foi desenvolvido e realizado:

- no prazo e no orçamento previstos;
- dentro das especificações técnicas e de qualidade previstas;
- cliente satisfeito com o produto ou serviço recebido.

Já os projetos malsucedidos apresentam problemas comuns, como, por exemplo:

- atrasos de cronograma;
- custos acima do previsto;
- falta de recursos humanos e materiais necessários;
- mudanças nos requisitos do projeto não controladas;
- qualidade do produto ou serviço abaixo da esperada;
- complexidade acima da capacidade da equipe;
- produtos mal projetados.

Para que um projeto atinja seus objetivos de forma eficaz, faz-se necessário um gerenciamento adequado. A ideia de gerenciamento de projetos iniciou com a definição de um guia genérico de boas práticas na gerência de projetos, o PMBOK (*Project Management Body of Knowledge*), com padrões e definições capazes de guiar a gerência em qualquer tipo de projeto. A partir disso, o PMBOK foi se tornando o principal e mais adotado modelo de gerenciamento de projetos, na área de desenvolvimento de software (PMI, 2004).

O gerenciamento de um projeto é composto por processos, por mais simples que seja o contexto ao qual se aplica. Com isso, boas práticas para a execução destes processos são necessárias para que se tenha um controle sobre todo o andamento do projeto em questão. Os grupos de processos são cinco etapas pelas quais qualquer projeto passará: iniciação, planejamento, execução, controle e encerramento (PMI, 2008).

A ***iniciação*** do projeto compreende dois processos, em que é desenvolvido o termo de abertura e são definidas as partes interessadas do projeto. Com isso, o escopo inicial pode ser definido e os recursos financeiros são acordados. O grupo de iniciação é muito importante para se definir qual a necessidade e onde se quer chegar com o desenrolar do projeto.

O grupo de ***planejamento*** é aquele que engloba o maior número de processos.

Isso se deve à necessidade de planejar muito bem o que será feito e como serão controladas as atividades que serão realizadas ao longo do projeto. É necessário definir com a maior precisão possível questões como escopo, tempo, custos e riscos do projeto, para que não ocorram surpresas e necessidades de replanejamento, já que o PMBOK sugere evitar, ao máximo, mudanças não previstas no planejamento.

O grupo de ***execução*** compreende processos que visam coordenar os recursos e atividades executadas, com o intuito de cumprir as necessidades definidas no projeto, em conformidade com o plano de gerenciamento realizado na etapa de planejamento. Este grupo compreende grande parte do orçamento do projeto e a execução dos processos pode evidenciar ou implicar mudanças, impactando em replanejamento, retrabalho e, possivelmente, atraso no projeto. Portanto, um gerenciamento eficaz da execução se faz necessário para o sucesso do projeto.

O grupo de processos de ***controle*** é composto por processos, os quais, como o nome já diz, definem práticas a serem executadas com o intuito de acompanhar e revisar o desempenho de projeto, validando se a execução de suas atividades está de acordo com o que foi planejado, monitorando recursos, mudanças e pontos críticos. São esses processos que proporcionam uma melhor visão de como está a saúde do projeto.

O grupo de processos de ***encerramento*** tem o intuito de finalizar o projeto, os quais definem boas práticas para se revisar o ciclo desenvolvido, obter aceitação do cliente, formalizar o término do projeto, encerrar as aquisições, documentar o desempenho do projeto e as lições aprendidas.

O quadro 6 apresenta um resumo dos grupos de processos da gerência de projetos.

Quadro 6 – Grupos de processo da gerência de projetos

Termo	Objetivo	Resultado
Iniciação	Definir o escopo inicial do projeto.	Termo de abertura
Planejamento	Planejar todas as atividades do projeto.	Plano do projeto
Execução	Executar as atividades de acordo como planejado.	Projeto executado
Controle	Acompanhar e revisar o desempenho do projeto.	Relatório de desempenho
Encerramento	Obter a aceitação do cliente.	Aceite do projeto

A divisão do projeto por áreas de conhecimento é uma outra perspectiva de organização de processos, baseada no contexto em que se aplica o processo e, não mais na sua relação com o ciclo de vida do projeto. Os processos estão distribuídos em nove áreas de conhecimento: escopo, tempo, integração, riscos, aquisições, custos, qualidade, comunicação e recursos humanos (PMI, 2008).

O ***escopo*** é o conjunto de requisitos necessários para entregar o projeto e seus produtos de acordo com as expectativas do cliente e partes interessadas. Um projeto deve ser entregue exatamente conforme o escopo, sem nenhuma funcionalidade a mais, pois a entrega de funcionalidades não previstas requer tempo e esforço desnecessários e não contratados.

A gerência de ***tempo*** é sinônimo de garantia da conclusão do projeto dentro do prazo predeterminado. Se o projeto foi entregue no prazo acordado, significa que, de alguma forma, a gerência de tempo foi bem-sucedida. O gerenciamento do tempo do projeto inicia com a definição das atividades necessárias para o projeto, bem como com as estimativas de esforço e duração para cada atividade. Com o sequenciamento e as estimativas em mão, o gerente do projeto inicia a definição de um cronograma das atividades, definindo suas datas de início e fim, datas de entregas dos pacotes e marcos de controle do andamento do projeto.

A gerência de ***integração***, como o próprio nome já sugere, consiste no planejamento e no controle de todas as atividades que visam integrar as fases do projeto. Portanto, gerenciar a integração nada mais é do que planejar e controlar, em uma perspectiva macro, os processos que englobam os cinco grupos definidos pelo PMBOK.

O princípio do gerenciamento de ***riscos*** consiste em identificar problemas que podem afetar o desempenho do projeto, mas que ainda não aconteceram. Como já visto neste livro, um risco é algo que, quando ocorre, tem impacto negativo no projeto. Um projeto, independente de sua área de atuação, sempre está sujeito a riscos, podendo ser técnicos, de ambiente, de recursos, entre outros, cada um com sua prioridade. A gerência de riscos atua em cima disso, procurando identificar os possíveis riscos na etapa de planejamento, já associando medidas e formas de lidar com os riscos que se concretizarem.

A gerência de ***aquisições*** dispõe de processos para garantir a aquisição e a administração de todos os recursos inerentes ao projeto e que não são supridos pela equipe do projeto. É uma área que atua diretamente com as necessidades já previstas para o projeto, planejando a aquisição como um todo, desde os fornecedores, critérios de avaliação de propostas, documentação contratual necessária para a aquisição dos

recursos e encerramento das aquisições, quando este encerramento se aplicar.

O gerenciamento de ***custos*** envolve todos os processos necessários para a estimativa e o controle dos custos para que a realização do projeto ocorra de acordo com o planejado. De acordo com o PMI (2008), a gerência de custos dispõe de processos que visam garantir com que o projeto se encerre dentro do orçamento que foi previsto e aprovado pelo cliente. Gerenciar custos envolve três atividades básicas: estimar, orçar e controlar.

Gerenciar a ***qualidade*** é atuar com processos que auxiliam para que as funcionalidades e a utilização do produto final estejam em conformidade com o que foi especificado para o projeto. A ausência ou má aplicabilidade do gerenciamento da qualidade impacta diretamente nas funcionalidades do projeto, satisfação do cliente, necessidade de retrabalho, motivação da equipe, eficiência e eficácia do produto. O gerenciamento da qualidade se dá pelo planejamento, garantia e controle da qualidade.

A ***comunicação*** é fundamental em um projeto, pois é um trabalho em equipe e requer que a informação chegue a todos os interessados de forma harmônica e rápida. A gerência de comunicação remete bastante ao papel do gerente de projetos, que é o responsável por facilitar a distribuição do fluxo de informações dentro do projeto.

O gerenciamento de ***recursos humanos*** é liderado pela figura do gerente de projetos, que procura garantir que as pessoas possuem condições de realizar o trabalho para o qual estão designadas. A gerência de recursos humanos se inicia com a definição dos papéis, responsabilidades, posição hierárquica e grau de conhecimento das pessoas para as atividades pelas quais são responsáveis. É dever da gerência de recursos humanos planejar o desenvolvimento pessoal de cada integrante do projeto, necessidades de treinamento, *feedbacks*, premiações e aspectos da vida pessoal que possam interferir no andamento do projeto. Outro aspecto que é analisado na gerência de recursos humanos envolve a motivação das pessoas, o que está diretamente ligado com a satisfação para com o ambiente de trabalho.

O quadro 7 apresenta um resumo das nove áreas de conhecimento da gerência de projetos.

Quadro 7 – Áreas de conhecimento da gerência de projetos

Área de conhecimento	Objetivo
Escopo	Definir e controlar o escopo do projeto.
Tempo	Garantir a conclusão do projeto dentro do prazo, previamente planejado e acordado.

Integração	Planejar e controlar todas as atividades que visam integrar as fases do projeto.
Riscos	Identificar e monitorar os possíveis problemas do projeto.
Aquisições	Garantir a aquisição e administração de todos os recursos necessários ao projeto.
Custos	Estimar e controlar os custos para a realização do projeto.
Qualidade	Planejar, controlar e garantir a qualidade do projeto.
Comunicação	Garantir que a informação chegue a todos os interessados de forma correta e rápida.
Recursos Humanos	Definir, acompanhar e motivar a equipe do projeto para a realização de um bom trabalho.

3.3.2 Gerência da configuração

Durante um projeto de software, muitos documentos, diagramas, especificações e códigos-fonte são gerados e constantemente modificados. Assim, dependendo da complexidade do projeto e do tamanho da equipe, a grande quantidade de material de trabalho produzido e as diversas mudanças nele podem levar o projeto ao caos. É comum, nestes casos, a não localização do material necessário, a incerteza de qual é a versão mais atual, dentre tantas outras dificuldades que acarretam retrabalho e atraso nos projetos.

Por isso, a gerência de configuração de software tem como objetivo estabelecer e manter a integridade de todos os produtos de trabalho de um projeto e disponibilizá-los aos envolvidos de forma rápida e confiável (CHRISSIS, 2003).

O gerente de configuração é o responsável por saber responder o que mudou, quando mudou, porque mudou e quem fez as mudanças nos produtos de trabalho de um projeto. Por isso, o desafio para uma boa gerência de configuração é encontrar o equilíbrio entre o excesso de burocracia, necessária para manter a integridade de tudo, e o controle adequado das mudanças, de forma a evitar problemas.

A gerência de configuração de software envolve controle de versão, controle de mudanças e integração contínua.

Em relação ao ***controle de mudanças***, deve-se registrar toda e qualquer mudança, tendo como informações o solicitante, a justificativa e o responsável.

Já quanto ao ***controle de versão***, todo o artefato deve ser rastreável no que tange a alterações realizadas, sempre registrando-se quem, quando e onde.

Quanto à ***integração contínua***, o objetivo é garantir que o sistema evolua de forma estável e funcional, podendo retornar sempre a uma configuração anterior.

Para que a gerência de configuração seja conduzida de forma adequada, é

necessária a elaboração de um plano de gerência de configuração, no qual se estabelecem normas, ferramentas e modelos que permitem gerenciar, de maneira satisfatória, os itens de configuração de um sistema.

Um item de configuração é a designação geral de qualquer artefato definido para ser mantido sob gerência de configuração. Já configuração, é uma coleção de revisões de itens de configuração que reunidas formam um módulo, subsistema ou produto.

Em cada fase do desenvolvimento de um software, um conjunto bem definido de itens de configuração deve ser estabelecido. A este conjunto é dado o nome de linha de base (do inglês *Baseline*) ou configuração base do sistema. Uma linha de base é uma "fotografia" da configuração do software em um dado momento do tempo, normalmente estabelecida em momentos estratégicos do projeto. Uma vez fechada, a linha de base não pode mais ser alterada.

3.3.3 Gerência da qualidade

São objetivos da gerência da qualidade definir explicitamente os critérios de qualidade, estabelecer as atividades necessárias para garanti-la, executar essas atividades nos projetos, utilizar indicadores para monitorar e melhorar a qualidade e fornecer relatórios sobre o uso do processo.

É importante destacar que não é papel da gerência da qualidade executar testes de software ou inspeção de documentação e demais itens do projeto, mas sim garantir que o processo de teste, bem como todos os demais, sejam realizados conforme a definição de cada organização, em todos os projetos.

A gerência da qualidade reduz a quantidade de trabalho repetido, permite reduzir custos, poupa tempo e aumenta a qualidade do produto final.

Segundo Pressmann (2001), um software de qualidade está em concordância com os requisitos e com os padrões de desenvolvimento explicitamente definidos e com as características implícitas em todo software desenvolvido profissionalmente.

As principais atividades da gerência da qualidade são:

- estabelecimento de um plano de garantia da qualidade;
- participação na definição do processo de software;
- revisão das atividades de engenharia de software para o seu ajuste ao processo;
- garantir que os desvios sejam documentados e geridos segundo o procedimento estabelecido;

- registrar o que não estiver ajustado aos requisitos e reportar à gerência superior.

O quadro 8 apresenta um resumo das atividades fundamentais do processo de software e o quadro 9 as atividades de apoio.

Quadro 8 – Atividades fundamentais do processo de software

Atividade	Objetivo	Resultado
Análise	Especificar as funcionalidades e restrições do sistema.	Especificação de requisitos do sistema
Projeto	Projetar a melhor estrutura para atender à especificação.	Diagramas técnicos
Implementação	Desenvolver, programar o sistema.	Sistema executável
Testes	Testar o sistema para identificação de falhas e não conformidades com a especificação.	Relatório de problemas
Implantação	Instalar e configurar o sistema, além de treinar os usuários.	Sistema pronto para o uso e usuários treinados
Manutenção	Corrigir defeitos, melhorar o sistema ou incluir novas funcionalidades.	Nova versão do sistema

Quadro 9 – Atividades de apoio do processo de software

Atividade	Objetivo	Resultado principal
Gerência de projeto	Planejar e controlar todas as atividades do projeto.	Plano de projeto e relatório de desempenho
Gerência de configuração	Controlar todos os artefatos do projeto.	Artefatos do projeto organizados e íntegros
Gerência de qualidade	Garantir que o processo definido está sendo seguido com qualidade.	Relatório de não conformidades

PARA COMPLEMENTAÇÃO DE ESTUDO

É importante destacar que este capítulo teve como objetivo apenas apresentar as principais atividades do processo de software, sem a pretensão de descrevê-las

completamente. Para aprofundar seus estudos sobre o processo de software consulte livros sobre engenharia de software como, por exemplo, Sommerville (2005) e Pressmann (2001). Além disso, uma ótima referência sobre o processo de software é o livro Managing The Software Process, do autor Watts Humphrey (HUMPHREY, 2004).

CAPÍTULO 4

PROCESSO DE ENGENHARIA DE REQUISITOS

Neste capítulo é apresentado o processo de engenharia de requisitos, bem como algumas das ferramentas de apoio disponíveis. Além disso, são apresentados alguns princípios para a melhoria desse processo nas organizações.

Conforme Kotonya e Sommerville (1998), o processo de engenharia de requisitos trata-se do conjunto de atividades necessárias para a elaboração e a manutenção do documento de requisitos de software.

Tal processo pode variar muito, desde um processo completamente não estruturado até um processo sistemático baseado na aplicação de métodos de análise. Os fatores que influenciam a variação do processo de requisitos são a maturidade técnica, disciplina e domínio da aplicação por parte da equipe, além da cultura organizacional.

Assim, cada empresa deve definir um processo próprio de engenharia de requisitos, o qual deve ser adaptado para a realidade de cada projeto. Entretanto, de uma forma geral, os diferentes processos de requisitos possuem cinco atividades fundamentais (Figura 12): elicitação, análise/negociação, documentação e validação e gerenciamento de requisitos.

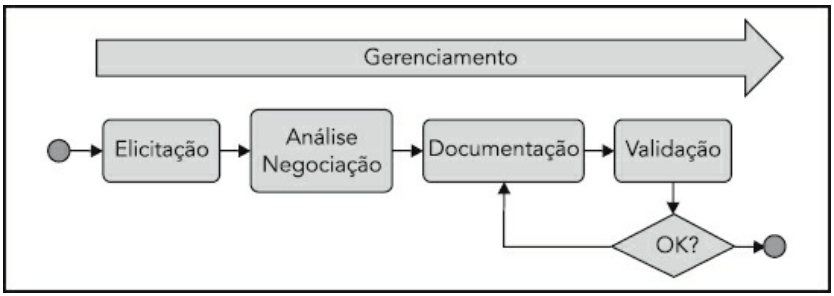

Figura 12 – Atividades da etapa de análise de um sistema.

Fonte: Autor.

4.1 Atividades

Conforme pode ser observado na figura 12, a primeira atividade do processo de engenharia de requisitos é a *elicitação de requisitos*, nome dado para a atividade de identificação, de descoberta dos requisitos necessários para o desenvolvimento de um sistema. Analistas trabalham junto com clientes e usuários para entender o problema e as necessidades e identificar o maior número possível de requisitos para o projeto.

Posteriormente, ou até mesmo em paralelo com a elicitação, o analista de sistema realiza a *análise* dos requisitos identificados com o objetivo de verificar se existem requisitos faltantes, requisitos desnecessários ou requisitos redundantes. Nesse caso, os problemas identificados pelo analista são negociados com o cliente a fim de se excluir os requisitos desnecessários e redundantes e incluir os requisitos faltantes, fechando-se, assim, o escopo inicial do projeto.

Somente após o fechamento do escopo, ou seja, da definição final da lista de requisitos do projeto, é que se inicia a *documentação* de tais requisitos. Nessa etapa, cada requisito é detalhadamente descrito de tal forma que o cliente possa validá-los e que a equipe de desenvolvimento possa compreendê-los a fim de construí-los da forma mais correta possível.

Assim que a documentação dos requisitos estiver concluída, inicia-se a etapa de *validação*, através da qual uma ou mais pessoas verificam o documento com o objetivo de certificar-se que cada requisito está claramente e corretamente descrito. Essa é uma etapa fundamental, visto que é através dela que o cliente verifica se a solução que o analista de sistemas elaborou para a sua necessidade está ou não de acordo. É através da validação dos requisitos que garantimos que o cliente saberá exatamente como é o sistema que receberá ao final do projeto.

Em paralelo com as atividades anteriores e durante todo o desenvolvimento e uso do sistema devemos realizar o *gerenciamento* dos requisitos. Essa atividade é fundamental para que toda e qualquer mudança necessária no sistema seja devidamente documentada e controlada, evitando-se que ao alterarmos um requisito outros sejam indevidamente afetados.

O quadro 10 apresenta um resumo das cinco principais atividades do processo de engenharia de requisitos.

Quadro 10 – Atividades da engenharia de requisitos

Atividade	Objetivo	Resultado principal
Elicitação	Identificar os requisitos do projeto.	Lista preliminar de requisitos
Análise/Negociação	Analisar a lista preliminar de requisitos e identificar requisitos desnecessários, requisitos faltantes e requisitos redundantes.	Lista definitiva de requisitos
Documentação	Especificar detalhadamente cada requisito da lista definitiva.	Documento de especificação de requisitos
Validação	Verificar a especificação de cada requisito para a identificação de problemas.	Lista de requisitos com problemas, bem como a descrição de cada problema identificado
Gerenciamento	Gerenciar mudanças na especificação dos requisitos.	Análise do impacto da mudança e nova versão da especificação

4.2 Ferramentas de apoio

Atualmente, existem diversas ferramentas de apoio ao processo de software, chamadas ferramentas CASE (do inglês *Computer-Aided Software Engineering*), que são softwares que apoiam o desenvolvimento de sistemas e o processo de evolução.

As ferramentas CASE proporcionam apoio ao processo de software pela automação de algumas atividades e pelo fornecimento de informações sobre o software que está sendo desenvolvido.

Através do uso de ferramentas CASE adequadas, pode-se otimizar os projetos de software, ou seja, é possível reduzir o tempo necessário para sua construção, diminuir os custos e aumentar a qualidade do produto final. Entretanto, como existem muitas ferramentas disponíveis, o desafio é saber escolher as ferramentas corretas. É preciso escolher a ferramenta certa para as características de cada organização, como o tipo de projeto, tamanho da equipe, tecnologias utilizadas, dentre outras.

As ferramentas CASE incluem editores de texto, gerenciadores de requisitos, editores de diagramas, compiladores, depuradores, gerenciadores de cronograma, gerenciadores de versões, gerenciadores de defeitos, automatizadores de testes, entre outras.

Quanto ao processo de engenharia de requisitos, existem dois tipos de ferramentas de apoio: ferramentas para modelagem e ferramentas para gerenciamento de requisitos.

Segundo Kotonya e Sommerville (1998), as características necessárias em uma ferramenta de apoio ao processo de requisitos são:

- navegador para visualização dos requisitos;
- sistema de busca por palavra-chave;
- suporte para rastreamento;
- geração de relatórios variados;
- integração com processadores de textos;
- controle de mudanças.

As três principais ferramentas, disponíveis atualmente, para o apoio ao processo de engenharia de requisitos são o Requisite Pro[1], ferramenta da IBM que integra um pacote que contém outras ferramentas de modelagem, testes e controle de versões, originalmente desenvolvido pela Rational Software; o CaliberRM[2], ferramenta da Borland, que é parte de uma suíte corporativa de produtos para engenharia de requisitos chamada Caliber Analist, e o DOORS[3], que faz parte de um pacote de ferramentas de gerenciamento e apoio ao desenvolvimento, desenvolvidas inicialmente pela Telelogic e, atualmente, comercializada pela IBM.

O Requisite Pro é a ferramenta para gerenciamento de requisitos e casos de uso da IBM, desenvolvida para equipes de projeto que desejam melhorar o entendimento dos objetivos do projeto, melhorar o desenvolvimento colaborativo, reduzir riscos e aumentar a qualidade das aplicações antes da entrega.

Essa ferramenta é fortemente integrada com o Microsoft Word, o que permite selecionar um texto e marcá-lo como requisito, gerando de forma automática seu identificador único. Além disso, no próprio Word é possível registrar palavras-chave para os requisitos e gerenciar suas propriedades.

O Requisite Pro também oferece documentos *templates* (de especificação de requisitos) que podem ser reutilizados na definição de outros projetos. Uma vez ingressados os requisitos, ele oferece facilidades para definir visões e propriedades. Além disso, as mudanças sobre as propriedades dos requisitos são registradas de forma automática. Ele fornece uma matriz de rastreabilidade bidirecional com indicadores visuais de mudanças nos artefatos do projeto (Figura 13), integração com Microsoft Project e Rational Rose e suporta bancos de dados Oracle, MS Access ou Microsoft SQL Server.

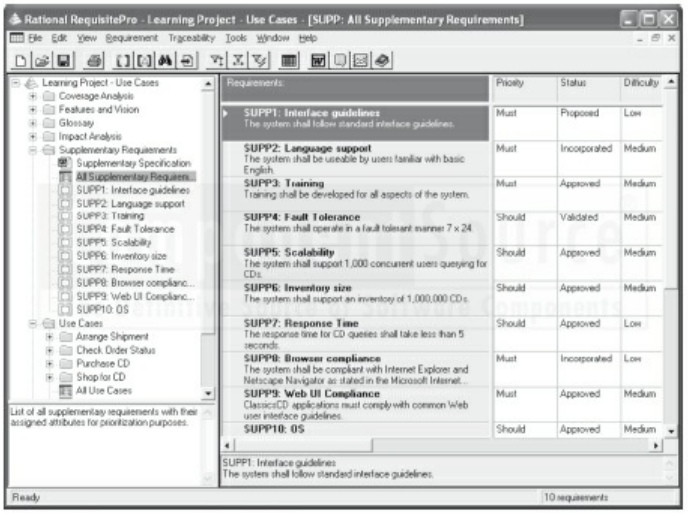

Figura 13 – Exemplo da interface do IBM Requisite Pro.
Fonte: http://www.ibm.com/software/awdtools/reqpro

Já o CaliberRM (Figura 14) é uma ferramenta de gerenciamento de requisitos da Borland, a qual é capaz de manter o rastreamento entre requisitos criados em um projeto e entre requisitos de projetos distintos.

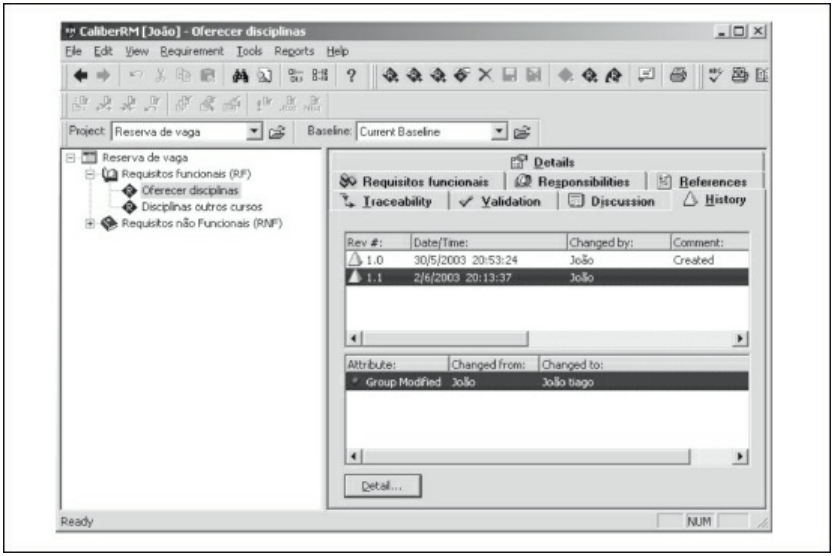

Figura 14 – Interface de manutenção de requisitos do CaliberRM
Fonte: http://www.ibm.com/software/awdtools/reqpro

Toda a rastreabilidade é mantida manualmente pelo usuário. Para se criar um relacionamento, um requisito é escolhido pelo usuário, em seguida a direção e o segundo requisito (ou artefato externo) são escolhidos.

Assim como no Requisite Pro, quando algum artefato é modificado dentro do CaliberRM, os relacionamentos que partem dele são marcados como "suspeitos", cabendo ao usuário analisar a mudança que foi feita e decidir se o relacionamento ainda é válido.

Já o DOORS (*Dynamic Object-Oriented Requirements System*) (Figura 15) é um pacote de ferramentas da IBM que visa gerenciar requisitos. O DOORS fornece *templates* para organizar e gerenciar as informações. Além disso, o DOORS inclui uma linguagem *script* para sua personalização ou desenvolvimento de extensões.

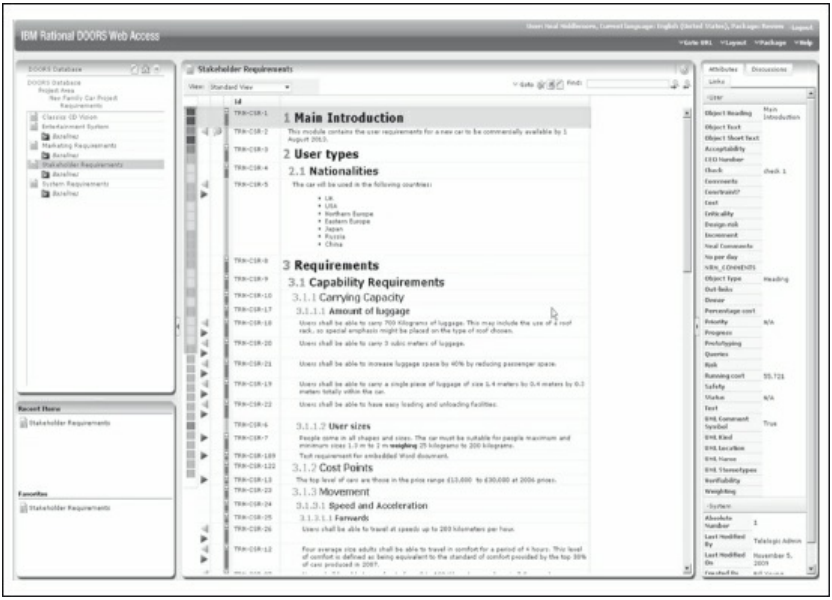

Figura 15 – Interface web do IMB DOORS.
Fonte: http://www.ibm.com/software/awdtools/reqpro

4.3 Melhoria do processo de requisitos

Segundo Humphrey (2004), a qualidade de um software está diretamente relacionada com a qualidade do processo utilizada na sua construção. Da mesma forma, a qualidade de uma especificação de requisitos depende muito da qualidade do processo utilizada na sua elaboração.

Assim, todo processo pode e deve ser melhorado continuamente, e os objetivos para melhoria são o aumento da qualidade do seu produto ou serviço, redução do tempo necessário para sua execução e redução dos recursos utilizados.

Como já citado no Capítulo 1, os principais problemas do processo de especificação de requisitos são a falta de envolvimento dos usuários, a não consideração das necessidades do negócio do cliente, a falta de gerenciamento dos requisitos, falta de definição de responsabilidades e, principalmente, problemas de comunicação com os usuários e a equipe de desenvolvimento.

Conforme Humphrey (2004), existem três níveis de maturidade para o processo de engenharia de requisitos: nível 1 (processo inicial), nível 2 (processo repetível) e nível 3 (processo definido).

O nível 1 de maturidade, considerado inicial, é caracterizado pela falta de um processo de especificação de requisitos definido, volatilidade excessiva dos requisitos, insatisfação dos usuários, muito retrabalho quando um requisito é alterado, documentação pobre e demorada e processo altamente dependente das pessoas que o executam.

Já no nível 2, chamado de repetível, o processo de requisitos já possui padronização, existem políticas para o gerenciamento dos requisitos, utiliza-se técnicas e ferramentas de apoio e a documentação é adequada e elaborada no prazo estipulado.

No nível 3 de maturidade, chamado de definido, o processo é baseado nas melhores práticas e existe um programa de melhoria contínua implantado.

PARA COMPLEMENTAÇÃO DE ESTUDO

Para aprofundar seu estudo sobre o processo de software consulte a referência Kotonya e Sommerville (1998) e Humphrey (2004). Já para complementar seu conhecimento sobre as ferramentas de apoio à engenharia de requisitos, consulte os sites dos fabricantes IBM (http://www.ibm.com), Telelogic (http://www.telelogic.com) e Borland (http://www.borland.com).

1. http://www.ibm.com/software/awdtools/reqpro
2. http://www.borland.com/br/products/caliber/rm.html
3. http://www.ibm.com/software/awdtools/doors

CAPÍTULO 5

ELICITAÇÃO DE REQUISITOS

Este capítulo detalha a primeira atividade da engenharia de requisitos, chamada de elicitação, e apresenta as pessoas envolvidas, as dificuldades comumente enfrentadas, as dimensões a serem consideradas e as técnicas para o processo de identificação dos requisitos em um projeto de software.

A elicitação de requisitos é nome dado para o conjunto de atividades que envolvem a descoberta dos requisitos de um sistema.

No processo de identificação de requisitos, analistas trabalham junto com clientes e usuários finais para entender o problema a ser resolvido através do detalhamento das funcionalidades e restrições do futuro sistema.

A elicitação de requisitos não se trata apenas de questionar os clientes e usuários sobre o que eles precisam, mas sim de uma análise da empresa, do domínio da aplicação e dos processos de negócio envolvidos.

Embora pareça ser uma atividade extremamente simples, a identificação dos requisitos envolve muitos pessoas e fatores subjetivos que dificultam muito os seus resultados. Entre os principais problemas para a adequada descoberta de requisitos podemos citar:

- as informações sobre o domínio da aplicação estão espalhadas em textos, manuais e na cabeça das pessoas que trabalham na área;
- as pessoas que entendem do problema a ser resolvido estão muito ocupadas tentando resolvê-los e não têm tempo ou não desejam ajudar o analista;
- diferentes interesses e políticas internas;
- os clientes geralmente não sabem o que precisam ou sabem mas não conseguem explicar;
- mudanças no negócio durante ou após a elicitação;
- mudanças no pessoal envolvido durante o processo.

5.1 Dimensões a serem consideradas

Conforme ilustrado na figura 16, são quatro as dimensões a serem consideradas no processo de identificação de requisitos.

Figura 16 – Dimensões do processo de elicitação de requisitos.
Fonte: Kotonya e Sommerville, 1998.

O *domínio da aplicação* refere-se ao conhecimento sobre a área de negócio para a qual o sistema será desenvolvido. O entendimento sobre o *problema* se refere ao detalhamento do problema específico do cliente em que o sistema será implantado. Quanto às *necessidades do cliente*, é necessária a compreensão em detalhe das necessidades específicas do cliente para que o sistema apoie seu trabalho. Quanto ao *contexto do negócio*, é necessário entender com clareza como o sistema afetará o negócio do cliente e como poderá contribuir para os objetivos do negócio.

Assim, a aceitação de um sistema depende do quanto e de como este atende as necessidades e expectativas dos usuários.

5.2 Processo de elicitação

A elicitação dos requisitos trata-se de um processo que envolve a identificação dos objetivos do cliente, compreensão do contexto do projeto e da organização para a qual o software será desenvolvido, além dos requisitos propriamente ditos.

A figura 17 apresenta as quatro etapas a serem consideradas para a adequada identificação dos requisitos em um projeto de software.

Figura 17 – Etapas do processo de identificação de requisitos.
Fonte: Kotonya e Sommerville, 1998.

Na etapa de *identificação dos objetivos* devemos compreender quais são as metas do negócio do cliente, para que o software a ser desenvolvido possa contribuir da melhor forma possível com os objetivos estratégicos do negócio. Além disso, mais do que saber quais funcionalidades o cliente deseja no software é preciso compreender exatamente qual é o problema que o cliente pretende resolver com o software a ser implantado. Nessa etapa, também é importante identificarmos quais são as restrições que o sistema a ser construído deve respeitar como, por exemplo, restrições tecnológicas, orçamentárias e de prazo.

Já na etapa de *entendimento do contexto* do projeto, é preciso compreender a estrutura da organização cliente, a fim de facilitar as negociações necessárias ao longo do processo de elicitação. Nesta etapa, também deve-se obter entendimento do domínio da aplicação, ou seja, é necessário obter informações prévias sobre a área de negócio para a qual o sistema será desenvolvido com o objetivo de facilitar a comunicação com os usuários e, assim, facilitar o processo de identificação dos

requisitos. Também, nesta etapa, deve-se conhecer os sistemas já existentes tanto na empresa do cliente como softwares já desenvolvidos para o mesmo fim. Dessa forma, é possível a identificação de requisitos de interface, integração e serviços a serem oferecidos pelo software a ser desenvolvido.

Na etapa de *organização*, deve-se identificar as pessoas as serem envolvidas e obter seus contatos, além de realizar a priorização dos objetivos do projeto, o que permite a adequada priorização dos requisitos identificados. Além disso, define-se exatamente qual é o escopo da organização para o qual o sistema será utilizado.

Finalmente, deve-se *identificar os requisitos* propriamente ditos através da interação com os usuários, do conhecimento sobre a área de negócio e da organização para a qual o sistema será desenvolvido.

5.3 Técnicas para elicitação de requisitos

Uma das técnicas mais utilizadas para a identificação de requisitos é a entrevista. Entretanto, essa técnica não é suficiente para a identificação do maior número possível de requisitos, que é um dos objetivos do processo de engenharia de requisitos.

Assim, cabe ao analista de sistemas conhecer as diferentes técnicas de elicitação existentes e saber quando aplicar cada uma delas. A escolha das técnicas e seu esquema de integração dependerão do problema e da equipe participante. O importante é conhecer as diferentes técnicas e identificar em que uma técnica é superior a outra.

A seguir são apresentadas as principais técnicas utilizadas na elicitação de requisitos de software.

5.3.1 Entrevistas

A entrevista, ou reunião com usuários, é uma técnica muito comum no processo de descoberta de requisitos de projetos de software. Através dela, o analista conversa sobre o sistema com diferentes pessoas e obtém um entendimento dos requisitos. Essa técnica apresenta como principal vantagem o contato direto com o usuário, o que proporciona negociação imediata. Entretanto, as diferenças de cultura e a incapacidade de muitos usuários para verbalizar sua forma de trabalho e de uma visão macro do negócio não permitem que todos os requisitos sejam identificados através dessa única técnica.

Existem três diferentes tipos de entrevistas: entrevistas abertas, as quais não possuem questões predefinidas; entrevistas fechadas, com questões predefinidas e tutorial, através da qual o usuário explica a sua forma de trabalho para o analista. É importante destacar que é possível aplicar uma combinação desses diferentes tipos.

Para a realização de boas entrevistas, o analista deve estar aberto para ouvir os envolvidos no projeto e mudar a sua opinião sobre o sistema, sempre que necessário. Assim, nunca devemos iniciar uma entrevista perguntando "o que você precisa?", mas sim através de uma pergunta, proposta de requisito ou discussão sobre algum sistema que o cliente já utiliza.

A técnica de entrevista é bastante eficiente para descoberta de requisitos mais genéricos, visto que os usuários gostam de falar de seu trabalho e dos problemas enfrentados. Entretanto, embora seja utilizada em todos os projetos, as entrevistas não são suficientes para elicitar todos os requisitos. Informações sobre domínio da aplicação não são suficientemente obtidas através de entrevistas, pois os usuários utilizam termos muito específicos para explicar suas tarefas e, muitas vezes, não conseguem explicar ou não acham que é necessário porque é óbvio.

5.3.2 Leitura de documentos

A técnica de leitura de documentos também é muito comum nos projetos de software, visto que é a forma como o analista frequentemente obtém ou complementa sua compreensão sobre o domínio da aplicação e sobre a organização do cliente.

É comum que os analistas façam um estudo prévio sobre a área de negócio para a qual o sistema deverá ser desenvolvido, a fim de facilitar a comunicação com os usuários e a compreensão do negócio. Para isso, é possível a leitura de livros, sites e artigos da área, bem como de material do próprio cliente, como manual de treinamentos, entre outros.

Além disso, como na maioria dos projetos os usuários não têm muita disponibilidade de tempo para a realização de reuniões, os analistas obtêm documentos utilizados pelos usuários, no seu dia a dia, para a realização de leituras com o objetivo de compreender os processos de trabalho para os quais o sistema deverá ser desenvolvido. Exemplos desse tipo de documento são formulários, relatórios, tutoriais, entre outros.

A vantagem dessa técnica é justamente o grande volume de informações disponíveis sem a dependência dos usuários. Entretanto, a técnica apresenta como desvantagens a impossibilidade de compreensão total a partir exclusivamente dos

documentos, dispersão das informações em diferentes locais e volume excessivo de trabalho, o que pode comprometer o prazo do projeto.

5.3.3 Questionários

A técnica de questionário trata-se de uma técnica através da qual o analista prepara um conjunto de questões prévias sobre os processos de trabalho dos usuários, bem como seus problemas e necessidades.

Esse técnica é recomendada para os casos em que o sistema a ser desenvolvido envolve muitos usuários, os quais estão dispersos em diferentes locais, e deseja-se obter um senso comum sobre suas necessidades e expectativas.

Assim, é possível obter uma ideia geral sobre como certos aspectos são percebidos, além da possiblidade da realização de análise estatística dos resultados.

A técnica apresenta como principal vantagem a padronização das perguntas e o tratamento estatístico das respostas, e como desvantagem a limitação do universo de respostas e pouca interação com os usuários.

5.3.4 Análise de protocolos

A análise de protocolo consiste em analisar o trabalho de determinada pessoa através de verbalização. O objetivo é estabelecer a racionalidade utilizada na execução de tarefas, ou seja, o usuário explica o que está fazendo durante a execução das tarefas.

Assim, a principal vantagem desta técnica é a possibilidade de elicitar fatos não facilmente observáveis e permitir melhor entendimento dos fatos. Como desvantagem destacam-se a possibilidade de o usuário não conseguir se expressar com clareza e o fato de que dificilmente consegue explicar exatamente como uma tarefa é executada.

5.3.5 Participação ativa dos usuários

Como o próprio nome sugere, essa técnica se dá através da incorporação dos usuários na equipe do projeto do software, ou seja, um ou mais usuários com bastante experiência nos processos e conhecimento da empresa cliente passam a integrar a equipe que está identificando e especificando os requisitos. Para isso, tais usuários precisam ser treinados e estar motivados a contribuir com o projeto.

A principal vantagem dessa técnica é a disponibilidade, em tempo integral, que o analista e a equipe do projeto têm para o esclarecimento de dúvidas e negociação das soluções propostas. Como desvantagem podem-se destacar a necessidade de treinamento dos usuários e a falsa impressão da eficácia do sistema, visto que todas as decisões são tomadas diretamente com um usuário-chave.

5.3.6 Observação

Como a maioria das pessoas, geralmente, acha difícil descrever o que faz, visto que isto é muito natural para elas, a melhor forma de entender o seu trabalho é observá-las enquanto o executam.

Através dessa técnica, o observador passa algum tempo observando as pessoas no trabalho e constrói uma imagem de como o trabalho é realizado.

A técnica de observação, muitas vezes, é complementar às entrevistas e análise de documentos, pois os processos reais de trabalho geralmente diferem das suas descrições formais e da forma como as pessoas os explicam.

Assim, a principal vantagem da técnica de observação é que ela proporciona uma visão mais completa e perfeitamente ajustada do contexto real de trabalho. Além disso, as pessoas preferem mostrar como fazem suas tarefas do que explicar o que fazem.

Como desvantagem, destaca-se o grande tempo gasto, a pouca sistematização do processo e o fato de que esta não é uma técnica apropriada para descoberta de requisitos organizacionais e de domínio, além do possível constrangimento causado nos usuários.

5.3.7 Reutilização de requisitos

O reuso envolve considerar requisitos que foram desenvolvidos para um sistema e usá-los em sistemas diferentes. Assim, é possível economizar tempo e esforço, pois requisitos reutilizados já foram analisados e validados em outros projetos.

Em muitos casos, o reuso de requisitos é um processo informal. Contudo, um reuso mais sistemático economizaria muito esforço e aumentaria a qualidade os produtos finais. Isso porque, na mesma área de aplicação, apenas 15% dos requisitos de um novo sistema são exclusivos dele. O restante são os mesmos de outros sistemas similares. Por isso, em todos os projetos de desenvolvimento de sistemas novos é

muito importante a pesquisa de sistemas similares.

Assim, a técnica de reutilização de requisitos apresenta como principal vantagem o aumento da produtividade e da qualidade dos sistemas, e como desvantagem a dificuldade de se promover reutilização sem modificação, o que gera a necessidade de toda uma nova validação dos requisitos modificados.

5.3.8 Cenários

Como as pessoas geralmente preferem visualizar algo do que obter a compreensão através de descrições abstratas, existem algumas técnicas que exploram os benefícios visuais para a obtenção dos requisitos em projetos de software. Exemplos dessas técnicas são os cenários, casos de uso e prototipação.

Através dessas técnicas, os usuários podem compreender e criticar, por meio de um diagrama, a proposta de solução para o sistema, e os analistas podem utilizar essas informações para formular os requisitos reais para o projeto.

Os cenários são histórias que explicam como um sistema poderá ser usado. Cada cenário aborda um ou um pequeno número de interações possíveis no sistema. Diferentes tipos de cenários foram desenvolvidos, e eles fornecem diferentes tipos de informações, com diferentes níveis de detalhes sobre o sistema. De modo geral, o cenário pode incluir:

- uma descrição do estado do sistema antes de começar o cenário;
- o fluxo normal de eventos do cenário;
- exceções ao fluxo normal de eventos;
- informações sobre atividades concorrentes;
- uma descrição do estado do sistema ao final do cenário.

A figura 18 apresenta um exemplo de um cenário de autenticação de um sistema.

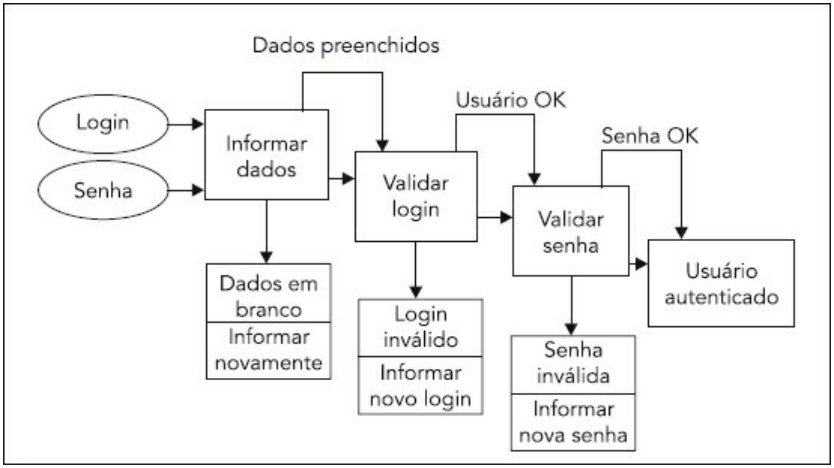

Figura 18 – Exemplo de um diagrama de cenário.
Fonte: Autor.

Como pode ser observado na figura 18, o diagrama de cenário apresenta o fluxo normal de interação, além das informações de entrada, condições de mudança de estado e possibilidades de exceção.

No exemplo, o usuário deve informar seu login e senha para realizar o processo de autenticação. Uma primeira possível exceção é a tentativa de autenticar sem informar os dados necessários. No caso de o usuário informar os dados necessários, o sistema passa para a próxima etapa que é a validação do login. Nesta etapa, a exceção possível é o login estar incorreto e o usuário terá que informar novamente. Finalmente, na última etapa, caso o login seja validado, o sistema realiza a validação da senha. Como na etapa anterior, neste caso a exceção possível é a senha estar inválida e, neste caso, o usuário deve informar a senha novamente. Caso a senha seja válida, o usuário estará autenticado no sistema.

5.3.9 Casos de uso

A técnica de casos de uso foi criada em 1993 por Jacobson e, atualmente, é parte fundamental da UML (*Unified Modeling Language*) para o desenvolvimento de sistemas orientados a objetos (SOMMERVILLE, 2005).

Um caso de uso ou *use-case* identifica os atores envolvidos em uma interação e especifica o tipo de interação. No diagrama de casos de uso os atores são representados por bonecos e cada caso de uso por uma elipse com um nome, o qual deve sempre ser iniciado por um verbo no infinitivo (JACOBSON, 1993).

Assim, um ator é algo ou alguém fora que interage com o sistema como, por exemplo, usuários, organizações, dispositivos físicos, sistemas externos ou componentes de um outro sistema. Já os casos de uso são, justamente, as possibilidades de interação que cada um dos atores tem com o sistema como, por exemplo, registro de um produto, consulta de um relatório, pesquisa de dados, entre outros.

A figura 19 apresenta um exemplo simples de um diagrama de casos de uso de um sistema bancário. Nesse exemplo, o cliente do banco pode utilizar o caixa automático para sacar dinheiro, transferir dinheiro ou consultar o saldo da conta. Assim, temos o cliente como um ator e as operações sacar dinheiro, transferir dinheiro e consultar saldo como casos de uso.

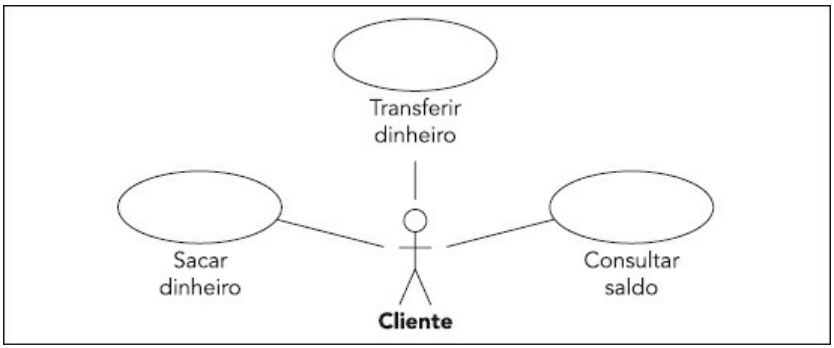

Figura 19 – Exemplo de um diagrama de casos de uso.
Fonte: Autor.

Cabe destacar que o diagrama é apenas uma forma visual de representar os atores e as possibilidades de interação de um sistema. A especificação detalhada de cada caso de uso é feita através de uma narrativa textual que pode incluir diversas informações específicas.

O objetivo aqui foi apresentar brevemente a técnica de casos de uso, a qual envolve uma série de outros conceitos e possibilidades que podem ser encontrados na

seguinte referência: Cockburn, Alistair. Escrevendo Casos de Uso Eficazes. Bookman. 2004.

5.3.10 Prototipação

A técnica de prototipação trata-se do processo de construir protótipos das telas do sistema com o objetivo de oferecer uma melhor compreensão da proposta do sistema tanto para o cliente como para a equipe de desenvolvimento.

Existem basicamente dois tipos de prototipação: descartável e reaproveitável. A prototipação descartável utiliza ferramentas de apoio que facilitam e aceleram o processo de criação de telas não funcionais do sistema que futuramente será construído. Neste caso, os protótipos só servem para o esclarecimento do sistema que está sendo proposto. Já a prototipação reaproveitável utiliza as próprias ferramentas de desenvolvimento de software para a criação prévia dos protótipos de telas, as quais futuramente serão aproveitadas para a construção do sistema. Neste caso, o tempo gasto com a prototipação é um pouco menor em função do reaproveitamento.

Contudo, apesar de ser uma técnica muito eficaz, a prototipação é bastante dispendiosa, mesmo com o uso de ferramentas de apoio e, consequentemente, consome muito tempo do projeto.

A figura 20 apresenta um exemplo de um protótipo de tela para um sistema.

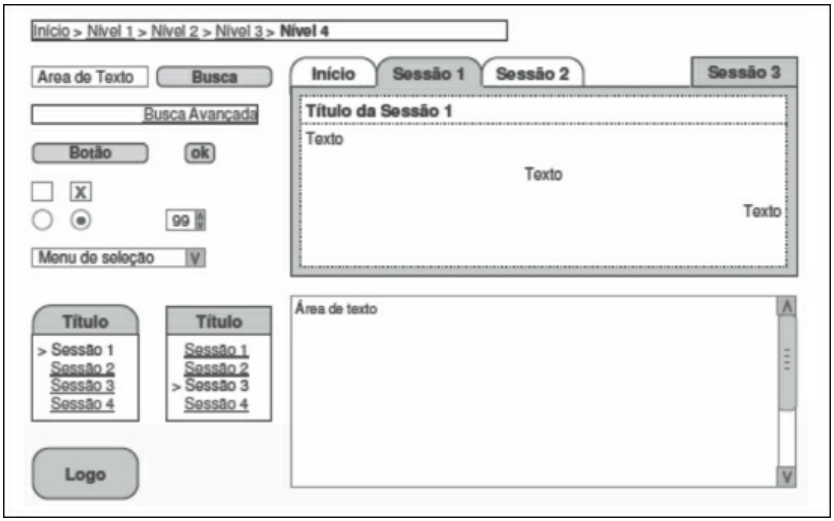

Figura 20 – Exemplos de um protótipo de tela.
Fonte: Frédéric Cavazza, http://iainstitute.org/tools

PARA COMPLEMENTAÇÃO DE ESTUDO

Complemente seus estudos sobre elicitação de requisitos consultando as seguintes referências: Kotonya e Sommerville (1998) e Humphrey (2004), Sommerville (2005), Cook-burn (2004) e Jacobson (1993).

CAPÍTULO 6

ANÁLISE E NEGOCIAÇÃO DE REQUISITOS

Este capítulo apresenta as etapas da análise e da negociação de requisitos, bem como conceitos e diferenças em relação às etapas de elicitação e validação. Além disso, são apresentadas algumas técnicas para a realização de uma adequada análise e negociação de requisitos em projetos de software.

A análise e a negociação são atividades que têm como objetivo descobrir problemas nos requisitos e obter um consenso na solução desses problemas junto aos clientes. A análise é feita após ou durante a elicitação dos requisitos e envolve a revisão de todos os requisitos com o objetivo de detectar problemas. Já a negociação pode ser feita para cada problema detectado, no caso da análise ser realizada em paralelo com a elicitação, ou para todos os problemas em conjunto se a negociação for feita após a elicitação e a análise.

A elicitação e a análise são processos que, geralmente, ocorrem em forma de espiral, ou seja, à medida que os requisitos vão sendo descobertos alguns problemas já vão sendo detectados e negociados com os clientes.

Já a análise e a validação de requisitos são atividades próximas, mas diferentes. A análise avalia os requisitos ainda incompletos e não aprovados pelos clientes e a validação inicia somente após os requisitos estarem documentados de forma detalhada e previamente aprovados pelos clientes.

6.1 Etapas da análise e negociação

Conforme ilustra a figura 21, a análise dos requisitos é composta da análise de necessidade, consistência, completude e de viabilidade. Já a negociação é realizada em três etapas, sendo elas a discussão dos problemas encontrados, priorização e aprovação dos requisitos.

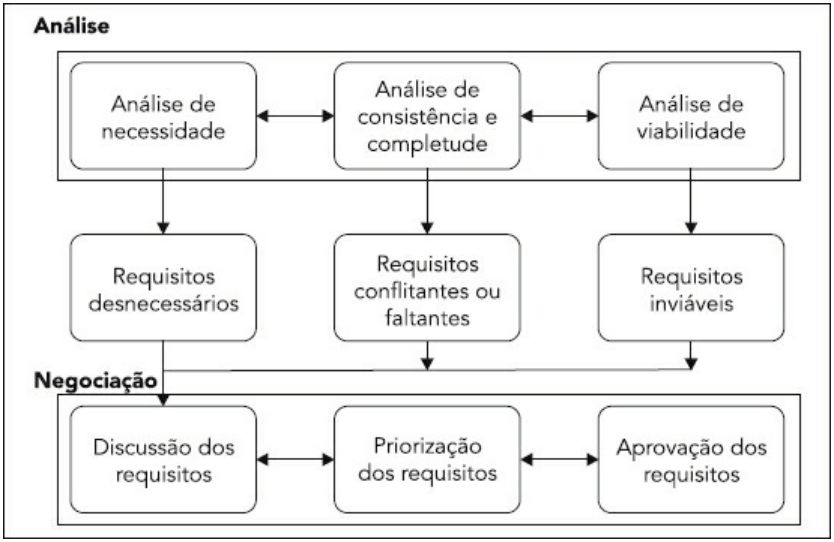

Figura 21 – Etapas da análise e da negociação.
Fonte: adaptado de Kotonya e Sommerville, 1998.

Na *análise de necessidade*, a real necessidade de cada requisito identificado é avaliada e uma lista de requisitos desnecessários, em relação ao objetivo do projeto e às necessidades do clientes, é gerada.

Já na *análise de consistência*, os requisitos são confrontados para evitar-se contradições. Quanto a *completude*, é realizada uma análise para verificar se todos os requisitos necessários foram identificados. Nessa etapa, uma lista de requisitos conflitantes e faltantes é gerada.

A análise de viabilidade tem como objetivo verificar a viabilidade dos requisitos que é verificada em relação a prazo, orçamento e tecnologia. Neste caso, uma lista de requisitos inviáveis é gerada.

Posteriormente, com base nas listas de requisitos desnecessários, conflitantes, inviáveis e faltantes, inicia-se a etapa de negociação, a qual se trata do processo de discutir os conflitos dos requisitos e estabelecer um consenso com os clientes.

A negociação inicia com a apresentação e a discussão dos problemas encontrados e, após, os requisitos são priorizados a fim de facilitar o processo de decisão. Finalmente, na etapa de aprovação, as soluções para os requisitos problemáticos são

acordadas entre os envolvidos e, assim, uma lista de requisitos para o projeto é definida.

Os principais pontos a serem considerados na negociação de requisitos de software são os objetivos da organização, necessidades dos usuários, restrições de implementação, orçamento e prazo. Já os fatores que normalmente influenciam as decisões são os interesses políticos, resistência dos usuários e a manipulação das pessoas envolvidas.

Apesar da etapa de análise e de negociação ser bastante eficaz, visto que evita que erros passem despercebidos, causando retrabalho no futuro, é muito comum e aceitável que durante o desenvolvimento do sistema novos problemas sejam identificados nos requisitos, bem como mais requisitos faltantes.

6.2 Técnicas para a análise

Existem diferentes técnicas para a realização da análise de requisitos de software, entre elas a técnica de revisão, *checklist* e matriz de interação.

6.2.1 Revisão

Trata-se de uma técnica de leitura individual da lista de requisitos, identificados durante a elicitação, com o propósito de encontrar defeitos. O termo "leitura" é usado para enfatizar as semelhanças com os processos mentais, utilizados para compreensão de qualquer texto. A técnica de revisão por leitura pode ser caracterizada como uma série de passos para o indivíduo analisar um produto textual, alcançando a compreensão precisa e identificando problemas (BERTINI, 2006).

6.2.2 Checklist

Um *checklist* trata-se de um conjunto de questões que o analista pode utilizar para analisar cada requisito com o objetivo de identificas problemas. O *checklist* pode ser construído como uma lista de questões, as quais devem ser respondidas para cada requisito identificado na elicitação, com a explicação detalhada de cada problema encontrado.

Os *checklists* são extremamente eficientes, pois tornam o processo mais rápido e

evitam esquecimentos. Conforme apresentado no quadro 11, um *checklist* para análise de requisitos deve conter questões para verificar se existem requisitos faltantes, desnecessários, conflitantes ou inviáveis.

Quadro 11 – *Checklist* para a análise de requisitos

Questões	Respostas
1. O requisito é realmente necessário para o objetivo do projeto?	
2. O requisito é realmente necessário para satisfazer as necessidades do usuário?	
3. O requisito está coerente com os demais requisitos já identificados?	
4. É possível desenvolver o requisito com a tecnologia disponível?	
5. É possível desenvolver o requisito no prazo necessário?	
6. É possível desenvolver o requisito com o orçamento disponível?	
7. Faltou algum requisito para que os objetivos do projeto sejam cumpridos?	
8. Faltou algum requisito para atender às necessidades dos usuários?	

6.2.3 Matriz de interação

Essa técnica possui o mesmo objetivo e é muito semelhante à de *checklist*. Entretanto, a matriz de interação permite coletar métricas a respeito da elicitação dos requisitos através da atribuição de pesos numéricos para problema encontrado, o que permite a identificação de pontos de melhoria no processo de engenharia de requisitos.

O quadro 12 apresenta um exemplo de uma matriz de interação para a análise de requisitos. Na última coluna, realiza-se o somatório dos pesos definidos para cada um dos possíveis problemas (questões) e, na última linha, o somatório dos pesos definidos para os problemas de cada requisito. Assim, ao final da análise podemos saber quais problemas foram mais frequentes e quais requisitos apresentaram mais problemas.

Quadro 12 – Matriz de interação para a análise de requisitos

Questões/Requisitos	RF01	RF02	RF03	RF04	Frequência dos problemas
1. O requisito é realmente necessário para o objetivo do projeto?					
2. O requisito é realmente necessário para satisfazer as necessidades do usuário?					
3. O requisito está coerente com os demais requisitos já identificados?					
4. É possível desenvolver o requisito com a tecnologia disponível?					
5. É possível desenvolver o requisito no prazo necessário?					
6. É possível desenvolver o requisito com o orçamento disponível?					
7. Faltou algum requisito para que os objetivos do projeto sejam cumpridos?					
8. Faltou algum requisito para atender às necessidades dos usuários?					
Problemas por requisios					

6.3 Técnicas para negociação

Assim como na análise, também existem diferentes técnicas para a realização de uma eficaz negociação dos problemas identificados nos requisitos, sendo a principal delas a realização de reunião.

Neste caso, a reunião deve ser realizada em três etapas. Inicialmente, é preciso informar todos os participantes de quais foram os problemas identificados. Depois, cada problema deve ser discutido e, finalmente, a solução para cada um deles deve ser tomada em conjunto.

PARA COMPLEMENTAÇÃO DE ESTUDO

Utilize as referências Kotonya e Sommerville (1998) e Bertini (2006) para complementar seu estudo sobre a etapa de análise e negociação de requisitos.

CAPÍTULO 7

DOCUMENTAÇÃO DE REQUISITOS

Este capítulo apresenta a importante atividade de documentação de requisitos, a qual gera um dos principais artefatos de um projeto de software, chamado de especificação de requisitos ou especificação funcional do sistema. Assim, são apresentados os diferentes públicos que utilizam esta documentação e os tipos de especificação, bem como as informações relevantes a serem documentadas e alguns padrões propostos.

A especificação de requisitos do software (ERS) é a declaração formal dos requisitos do sistema e pode conter apenas os requisitos de usuário ou os de usuário e os de sistema.

A ERS é utilizada por diferentes públicos. Os clientes utilizam a ERS para verificar se a solução proposta pelo analista ficou de fato clara e coerente com as suas necessidades e expectativas. Já os gerentes a utilizam para definir o escopo do projeto e, consequentemente, estimar equipe, prazos e o orçamento. Os projetistas utilizam para compreender a proposta do sistema e definir a melhor estrutura que atenda à especificação. Testadores utilizam a documentação para planejar previamente os testes e para verificar se o sistema construído está em conformidade com o especificado (KOTONYA e SOMMERVILLE, 1998).

Além disso, a documentação dos requisitos também é extremamente útil nos casos de manutenção do sistema, visto que é através dela que se pode obter entendimento de quais são os serviços oferecidos pelo sistema, além da relação entre eles.

7.1 Informações relevantes

Existem diferentes padrões para a documentação de requisitos de software, e cada empresa deve definir o tipo de documentação que é mais adequada às suas necessidades e características.

Contudo, existem algumas informações que são importantes de serem registradas nos diferentes tipos de documentação. São elas:

- versão do documento;
- nome do autor;
- data de criação do documento;
- nome do projeto;
- objetivo do projeto;
- requisitos funcionais;
- requisitos não funcionais;
- glossário de termos.

O controle de versões do documento de especificação de requisitos é extremamente importante visto que este documento é utilizado por muitas pessoas e está em constante atualização. Somente com um adequado versionamento do documento é possível garantir a sua integridade, de tal forma que todos os envolvidos saibam quem, o que e quando foi alterado.

O quadro 13 ilustra uma forma simples de manter o controle de versões de um documento. Assim, para cada alteração são registrados a nova versão do documento, além da data, autor e motivo da alteração. É importante destacar que existem ferramentas de apoio que automatizam e garantem o versionamento tanto de requisitos como de documentos.

Quadro 13 – Exemplo simples de controle de versão de documentação

Versão	Data	Descrição	Autor
1.0	17/9/2010	Primeira versão do documento	Analista 01
1.1	18/9/2010	Inclusão de novos requisitos funcionais (RF045, RF 046)	Analista 02
1.2	18/9/2010	Complementação do RF 035	Analista 01
1.3	20/9/2010	Exclusão do RF 010	Analista 01

As informações documentadas sobre os requisitos variam conforme o padrão de cada organização, mas, de forma geral, registra-se um identificador único para cada

requisito, o nome e a descrição detalhada, além de alguns atributos como prioridade, complexidade, situação, autor e data de criação.

O quadro 14 apresenta um exemplo de documentação de um requisito funcional. Trata-se da especificação da funcionalidade para submissão de artigos em um sistema de gerenciamento de congressos. Como pode ser observado, são registrados o identificador do requisito (RF023), seu nome, descrição detalhada com as regras de negócio a serem respeitadas, além dos atributos.

Quadro 14 – Exemplo de documentação de um requisito funcional

RF 023 – Submissão de artigos

O sistema deverá possibilitar, para usuários autenticados, a submissão de artigos para serem avaliados, solicitando, obrigatoriamente, as seguintes informações:

- Nome do autor principal;
- e-mail do autor principal;
- coautores do artigo;
- título do artigo;
- resumo do artigo;
- arquivo contendo o artigo.

As seguintes regras de negócio devem ser respeitadas para a submissão de artigos:

- O nome e o e-mail do autor principal, além do título, resumo e arquivo contendo o artigo são informações obrigatórias;
- o e-mail deve ser um endereço válido;
- o arquivo contendo o artigo deve estar no formato PDF;
- esse serviço só deve estar disponível até a data limite definida para a submissão de artigos de cada congresso.

Prioridade	Complexidade	Status	Versão	Autor
Alta	Baixa	Aprovado	1.0	Analista 02

A prioridade, que pode ser baixa, média ou alta, tem como objetivo identificar os requisitos mais importantes, os quais devem ser cuidadosamente projetados, construídos e testados. Ela também auxilia na definição da ordem em que os requisitos serão construídos.

Já a complexidade, que também pode ser baixa, média ou alta, tem como objetivo auxiliar nas estimativas de esforço e de complexidade do sistema como um todo, além de contribuir para a distribuição dos requisitos entre os programadores, sendo que os mais experientes devem desenvolver os requisitos mais complexos.

O registro da situação do requisito é importante para informar todos os envolvidos no projeto se o requisito está sendo elaborado, se está em avaliação pelo cliente, se já está aprovado pelo cliente ou se foi excluído do projeto. Mesmo que a equipe decida pela exclusão de um requisito já documentado, esse requisito não deve ser de fato apagado da especificação, visto que em algum momento posterior do projeto essa decisão poderá ser revogada. Neste caso, deve-se apenas alterar o atributo situação para excluído.

Além disso, como em um mesmo projeto, dependendo de sua complexidade e tamanho, pode-se ter mais de um analista especificando requisitos, é importante registrar a data e o nome do analista que criou cada requisito.

O quadro 15 apresenta um exemplo de documentação de um requisito não funcional de produto, categoria portabilidade.

Quadro 15 – Exemplo de documentação de um requisito não funcional

RNF003 – Compatibilidade com navegadores de internet

O sistema deverá suportar os seguintes navegadores de internet: Inter Explorer 7 ou superior, Mozilla Firefox 3.6 ou superior. Google Chrome 1.0 ou superior e Safari 5.0 ou superior.

Prioridade	Complexidade	Status	Versão	Autor
Alta	Média	Avaliação	1.0	Analista 01

O glossário de termos tem como objetivo auxiliar a compreensão de todos os

envolvidos sobre os requisitos especificados. Assim, o glossário deve conter definições para todos os termos específicos do negócio e termos técnicos, além das siglas e abreviações.

O quadro 16 ilustra um exemplo de definições para um glossário de termos.

Quadro 16 – Exemplo de parte de um glossário de termos

A

- **Autor:** pessoa que realizou a elaboração do artigo.
- **Avaliador:** pessoa que realiza as avaliações dos artigos submetidos.

B

- **Browser:** navegador de páginas da internet como, por exemplo, o Internet Explorer.

C

- **Coautor:** pessoa que contribui na elaboração do artigo.
- **Criptografia:** técnica utilizada para garantir sigilo das informações, através da sua transformação de modo original para outro ilegível.

7.2 Padrões propostos

Com o passar do tempo, alguns padrões para a documentação de requisitos foram propostos. Entre eles, podemos citar o padrão proposto pela IEEE (*Institute of Electrical and Electronics Engineers*) e o proposto pelo RUP (*Rational Unified Process*).

7.2.1 IEEE 830-1998

O padrão IEEE 830 (1998) sugere um estrutura para a organização do documento

de especificação de requisitos, conforme ilustra o quadro 17.

Quadro 17 – Proposta de estrutura para documentação de requisitos

1. Introdução
- 1.1 Propósito do documento
- 1.2 Escopo do sistema
- 1.3 Definições, acrônimos e abreviaturas
- 1.4 Referências
- 1.5 Visão geral do documento

2. Descrição geral
- 2.1 Perspectiva do produto
- 2.2 Funções do produto
- 2.3 Características dos usuários
- 2.4 Restrições gerais
- 2.5 Premissas e dependências

3. Requisitos específicos

Apêndices

Índice

Fonte: IEEE 830-1998.

Conforme pode ser observado na figura, o documento deve ser composto de três partes, sendo elas introdução, descrição geral e requisitos específicos, além dos apêndices e índice.

Introdução

A introdução deve fornecer uma visão geral do documento todo e está organizada da seguinte forma:

- propósito do documento;

- escopo do sistema;
- definições, acrônimos e abreviaturas;
- referências;
- visão geral do documento.

A primeira parte da introdução trata, justamente, de informar o leitor qual é o propósito do documento de especificação de requisitos. Essa parte também pode conter o público a que se destina o documento e descrever seu histórico de versão e um sumário das mudanças feitas em cada versão.

Já no item relacionado ao escopo do sistema, deve-se identificar o produto de software ao qual se refere a documentação, descrever brevemente o sistema contribuirá para o negócio, ou seja, definir como o sistema afetará o negócio e como ele poderá contribuir para os objetivos do negócio. Além disso, deve-se definir a abrangência do sistema, declarando brevemente o que fará e o que não fará parte do mesmo.

Outro item que faz parte da introdução refere-se aos termos, siglas e abreviações, os quais têm como objetivo permitir a correta interpretação da documentação. Já a seção referências tem como objetivo informar outros documentos citados na especificação dos requisitos, através de seu título e da data de elaboração.

A última parte da introdução tem como objetivo informar quais são as informações contidas na documentação e como essas informações estão organizadas.

Descrição geral

Esta seção descreve os fatores gerais que afetam o sistema e seus requisitos. Essas informações têm como objetivo fornecer um contexto e facilitar a compreensão dos requisitos e estão organizadas da seguinte forma:

- perspectiva do produto;
- funções do produto;
- características dos usuários;
- restrições gerais;
- premissas e dependências.

A primeira parte da descrição geral trata da definição do propósito do produto, através da qual define-se se o produto será utilizado de forma isolada, se fará parte de um sistema maior ou se será integrado com outros sistemas.

Na parte que trata das funções do produto, deve-se informar resumidamente

quais são as principais funcionalidades que o sistema fornecerá. Para isso, é possível utilizar apenas texto ou diagramas simples ilustrando as principais funções do sistema e as relações entre elas.

Quanto às características dos usuários, é necessário documentar informações gerais que permitam a compreensão de qual é o perfil das pessoas que futuramente utilizarão o sistema. Assim, documenta-se informações como o nível de educação dos diferentes grupos de usuários, bem como suas experiências com o uso de software e tecnologias.

Já na parte das restrições, registra-se todas as limitações impostas ao sistema como, por exemplo, limitações de hardware, interfaces com outros sistemas, acessibilidade, segurança, políticas internas do cliente, entre outras.

A última parte da descrição geral do produto trata das premissas e dependências, através das quais listam-se todos os fatores que afetam os requisitos do sistema como, por exemplo, a disponibilidade de uma tecnologia ou servidor no cliente que influencia na definição dos requisitos.

Requisitos específicos

Nesta seção todos os requisitos funcionais e não funcionais do produto devem ser detalhadamente especificados, de tal forma que os clientes possam verificar se o sistema proposto atende suas necessidades e expectativas. Além disso, a descrição dos requisitos deve permitir que os projetistas compreendam o sistema a ser projetado, os desenvolvedores saibam como implementar cada funcionalidade e os testadores saibam como validar o sistema.

Apêndices e índice

Os apêndices contêm informações complementares para a compreensão do documento de especificação como, por exemplo, formulários e relatórios atualmente utilizados pelo cliente, dentre outros. Já o índice tem como objetivo facilitar a localização das informações contidas na documentação.

7.2.2 RUP

O padrão para documentação de requisitos do *Rational Unified Process* (EVANS, 2003) está organizado em cindo diferentes documentos:

- plano de gerenciamento de requisitos;
- documento de visão;
- requisitos suplementares;
- casos de uso;
- glossário.

Para cada um dos documentos propostos existe um modelo contendo os itens a serem preenchidos, bem como uma explicação sobre como ser o seu preenchimento. É importante destacar que cada empresa que utiliza o padrão RUP deve adaptar os modelos de documentos, conforme as suas características, bem como em função das características de cada projeto.

O *plano de gerenciamento de requisitos* trata das definições gerais sobre como os requisitos serão organizados, identificados e documentados. Além disso, registram-se nesse documento informações sobre os envolvidos no projeto, tanto da parte do cliente como da equipe de desenvolvimento.

Já o *documento de visão* descreve o problema a ser resolvido, os objetivos e premissas do projeto, bem como os requisitos funcionais do sistema.

No documento de *requisitos suplementares* são definidos os requisitos não funcionais do sistema, os quais são organizados nas seguintes categorias: usabilidade, confiabilidade, desempenho, suportabilidade, restrições de projeto, documentação de usuário, interface, licenciamento, legislação e padrões.

Como o RUP é um modelo para o desenvolvimento de software orientado a objetos, ele propõe a identificação e documentação dos *casos de uso* do sistema. Assim, cada um dos casos de uso deve possuir um documento contendo as seguintes informações: nome do caso de uso, descrição breve, fluxo de eventos (base e alternativos), requisitos especiais relacionados, precondições, pós-condições pontos de extensão.

Finalmente, no *glossário* de termos são definidas as siglas, as abreviações e os termos técnicos e de negócio necessários para a adequada compreensão da documentação.

PARA COMPLEMENTAÇÃO DE ESTUDO

Consulte as referências Kotonya e Sommerville (1998), IEEE 830 (1998) e Evans

(2003) para complementar seu estudo sobre padrões para documentação de requisitos de software. Além disso, você pode acessar o site da IBM (http://www.ibm.com) e da IEEE (http://www.ieee.org).

CAPÍTULO 8

VALIDAÇÃO DE REQUISITOS

Este capítulo apresenta a essencial etapa de validação de requisitos, seu objetivo e os envolvidos, bem como o processo de validação e as técnicas mais utilizadas.

Conforme a IEEE 830-1998, uma especificação de requisitos deve descrever os requisitos de forma clara, precisa, completa e consistente. Por isso, a validação tem como objetivo verificar, pelo menos, a clareza, ambiguidade, completude e consistência dos requisitos, a partir do documento de especificação de requisitos do software.

Um requisito descrito de forma clara é fácil de compreender. Já um requisito ambíguo possibilita diferentes formas de compreensão, o que pode levar o desenvolver ou o testar a interpretá-lo de forma errada, causando prejuízo para o projeto.

Um outro problema comumente identificado na etapa de validação é a inconsistência entre requisitos do projeto, ou seja, dois requisitos que apresentam algum grau de dependência, estão definidos de forma incompatível.

Também é muito frequente encontrarmos requisitos descritos de forma incompleta na documentação, o que dificulta ou até mesmo atrasa o desenvolvimento do sistema, já que tais informações terão que ser obtidas futuramente.

Além disso, outros problemas podem ser identificados na etapa de validação, como a não conformidade com o padrão de documentação definido pela empresa desenvolvedora do software, a especificação de mais de um requisito como se fosse uma única funcionalidade, o que chamamos de combinação de requisitos, entre outros.

8.1 Processo de validação

O processo de validação de requisitos trata-se do conjunto de atividades realizadas para identificação de problemas na especificação de requisitos de software

(ERS). Conforme ilustra a figura 22, a partir do conhecimento da organização e das necessidades do cliente e dos padrões para documentação de requisitos, realiza-se a validação da ERS e gera-se um lista de problemas para os quais se deve elaborar um plano de ação.

Figura 22 – Processo de validação de requisitos.
Fonte: adaptado de Kotonya e Sommerville,1998.

Clientes, analistas de sistema, engenheiros de software e testadores devem, na medida do possível, validar a documentação de requisitos, a fim de identificar problemas que possam ser corrigidos previamente.

Essa atividade é de extrema importância, visto que é através dela que os clientes podem acordar com a empresa, fornecedora do software, como deve ser o sistema que será desenvolvido, evitando surpresas e frustrações tanto para o cliente como para o fornecedor.

8.2 Custo das falhas

Estudos provaram que o custo para correção de um requisito aumenta, consideravelmente, à medida que o projeto avança. Assim, o momento mais adequado para identificar e corrigir problemas é, de fato, durante a própria especificação dos requisitos.

Leffingwell e Widrig (2003) apontam qual o custo para correção de um problema, de acordo com a fase do processo de software em que foi encontrado (Figura 23), e afirmam que, se uma unidade de custo é o esforço necessário para corrigir um erro durante o estágio de codificação, então o custo necessário para corrigir um erro durante o estágio de engenharia de requisitos é de 5 a 10 vezes menor. Já o custo necessário para corrigir um erro durante o estágio de manutenção é 20 vezes maior.

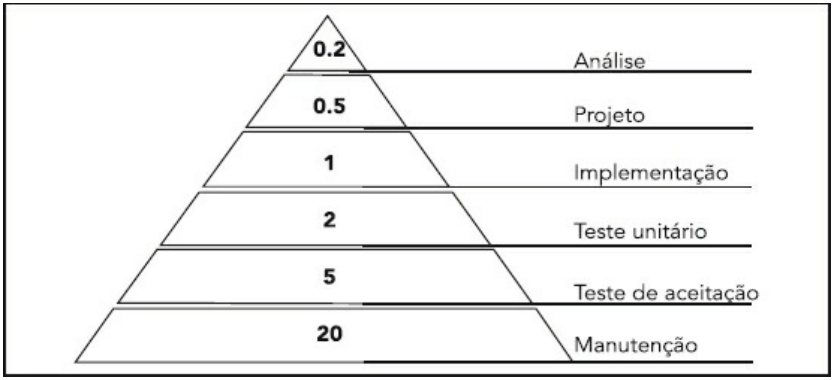

Figura 23 – Custo de erros em projetos de software.
Fonte: LEFFINGWELL e WIDRIG, 2003.

8.3 Técnicas para validação de requisitos

As técnicas utilizadas para a validação de requisitos são as mesmas utilizadas na etapa de análise, porém com o objetivo de validar diferentes aspectos.

Na etapa da análise, o objetivo é verificar se os requisitos corretos foram identificados. Já na etapa de validação, o objetivo é verificar se os requisitos identificados foram corretamente documentados.

8.3.1 Revisão

A revisão, cujas etapas são apresentadas na figura 24, é a técnica mais comum na validação de requisitos e trata-se da revisão da especificação por um grupo de pessoas, as quais se reúnem para discutir os problemas encontrados e definir soluções para os mesmos.

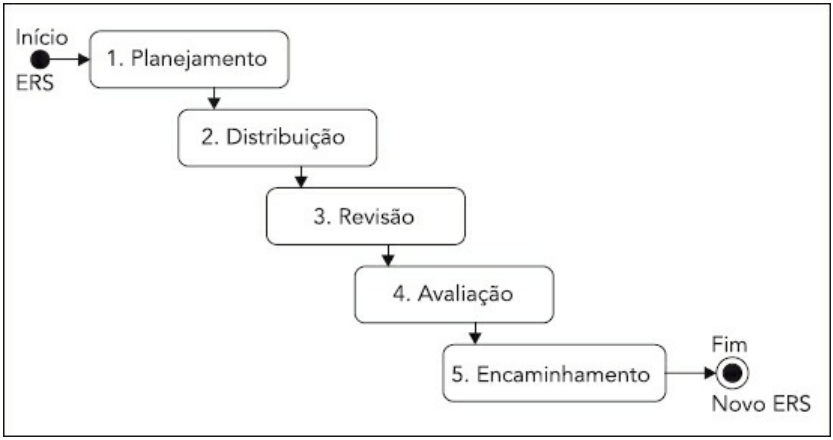

Figura 24 – Etapas da revisão de requisitos.
Fonte: adaptado de Kotony e Sommerville, 1998.

O processo de revisão inicia como planejamento de quais são as pessoas que irão participar, do prazo a ser disponibilizado e da data, local e horário da reunião para a discussão dos problemas identificados. Posteriormente, a ERS é distribuída às pessoas que realizarão a revisão individual. Finalmente, é feita uma reunião para avaliação dos resultados da revisão e encaminhamento dos problemas identificados.

As possíveis ações para cada problema são a reescrita, através da qual o autor do requisito deve reescrevê-lo a fim de melhorar a sua descrição; a complementação, na qual o autor acrescenta informações; a correção, através da qual erros e conflitos são resolvidos, e a exclusão, no caso de o requisito ser considerado inviável ou desnecessário. Após a correção dos problemas, uma nova versão da especificação de requisitos é gerada, a qual será encaminhada para as próximas etapas do projeto.

8.3.2 Checklist

Como já discutido neste livro, um *checklist* trata-se de um conjunto de questões utilizadas para validar cada requisito da especificação.

Conforme apresentado no quadro 18, um *checklist* para validação de requisitos deve conter questões para verificar, pelo menos, se os requisitos estão claros, completos, precisos e consistentes, além de verificar se cada requisito especifica exclusivamente uma funcionalidade.

Quadro 18 – *Checklist* para validação de requisitos

Questões	Respostas
1. A descrição do requisito está clara?	
2. Todas as informações necessárias foram especificadas?	
3. Existe apenas uma forma de compreender o requisito?	
4. O requisito apresenta alguma incompatibilidade com os demais?	
5. O requisito descreve apenas uma funcionalidade?	

8.3.3 Matriz de interação

Assim como na análise, a matriz de interação permite coletar métricas a respeito da validação dos requisitos através da atribuição de pesos numéricos para problema encontrado, o que permite a identificação de pontos de melhoria no processo de documentação dos requisitos.

No quadro 19 apresenta um exemplo de matriz de interação para validação de requisitos. Na última coluna, realiza-se o somatório dos pesos definidos para cada um dos critérios avaliados e, na última linha, o somatório dos pesos definidos para os problemas de cada requisito. Assim, ao final da validação podemos saber quais problemas foram mais frequentes e quais requisitos apresentaram mais problemas.

Quadro 19 – Matriz de interação para validação de requisitos

Créditos/Requisitos	RF01	RF02	RF03	RF04	Frequência dos problemas
Clareza					
Ambiguidade					
Completude					
Consistência					
Combinação					
Problemas por requisitos					

Os pesos para cada critério devem ser definidos por cada organização. De forma

geral, definem-se as seguintes opções:

- 00: nenhum problema;
- 01: problema leve;
- 02: problema moderado;
- 03: problema grave.

Tão ou mais importante que definir pesos para os problemas encontrados é a descrição detalhada de cada problema, o que permite que sejam rapidamente compreendidos e corrigidos. O quadro 20 apresenta um exemplo de formato para detalhamento dos problemas encontrados.

Quadro 20 – Relatório de problemas encontrados

#	Requisito	Categoria	Descrição
1	RF024	Clareza	Não está claro quais são os perfis de usuário que devem ter acesso à funcionalidade.
2	RF027	Consistência	Não é possível gerar a informação sobre a faixa etária dos clientes, visto que no cadastro de clientes não é registrada a data de nascimento.
3	RF047	Combinação	Existem duas funcionalidades especificadas neste requisito funcional.
4	RNF01	Ambiguidade	Não é possível ter certeza de que o sistema deverá funcionar em qualquer versão do navegador Firefox ou apenas nas versões atual e superiores.
5	RNF04	Completude	Faltou especificar quais são as versões dos sistemas operacionais em que o software será utilizado.

Além das técnicas de revisão, *checklist* e matriz de interação, também podem ser desenvolvidos o manual de usuário e o plano de testes, a partir da especificação dos requisitos, como forma de validá-los.

PARA COMPLEMENTAÇÃO DE ESTUDO

Utilize as referências Kotonya e Sommerville (1998), Sommerville (2005), IEEE 830 (1998) e Leffingwell e Widrig (2003) para complementar seu estudo sobre a etapa de validação de requisitos de software.

CAPÍTULO 9

GERENCIAMENTO DE REQUISITOS

Este capítulo apresenta a atividade de gerenciamento de requisitos, a qual tem como objetivo principal garantir a integridade da especificação, sempre que for necessária alguma alteração. Assim, o capítulo trata das necessidades e dos benefícios do gerenciamento de requisitos, bem como apresenta as diferentes possibilidades de identificá-los e armazená-los. Além disso, é apresentado o processo de gerenciamento de mudanças e a rastreabilidade de requisitos.

Visto que ocorrem mudanças nos requisitos, em todas as etapas do processo de software, além do surgimento de novos, o gerenciamento de requisitos tem como objetivo gerenciar todas essas mudanças, a fim de minimizar suas possíveis consequências.

Como já citado anteriormente, os principais motivos para mudanças nos requisitos de projetos de software são erros, conflitos, inconsistências, evolução no entendimento, problemas ou restrições técnicas, prazo, orçamento, além de mudanças nas prioridades do cliente, no negócio, no processo e na equipe envolvida.

9.1 Gerenciamento de mudanças

O gerenciamento de todas as mudanças que, inevitavelmente, ocorrem nos requisitos de um software, durante seu desenvolvimento e uso, é extremamente importante para garantir a integridade do sistema. Para garantir um adequado gerenciamento das mudanças, as seguintes definições são necessárias:

- definição de um processo para requisição de mudanças, bem como das informações necessárias para cada requisição;
- processo para analisar o impacto e os custos das mudanças;
- equipe que irá avaliar as propostas de mudança;

- software a ser utilizado para gerenciar as mudanças.

Conforme ilustrado na figura 25, o processo de mudança inicia a partir do recebimento de uma solicitação de mudança, a qual é validada e, se for aceita, os requisitos afetados direta e indiretamente são identificados. Após a análise de impacto, uma proposta de mudança é elaborada e avaliada. Caso a proposta seja aceita, ela só será de fato executada se o seu orçamento for previamente aprovado. Caso contrário, uma nova proposta de solução, com custo menor, deve ser elaborada.

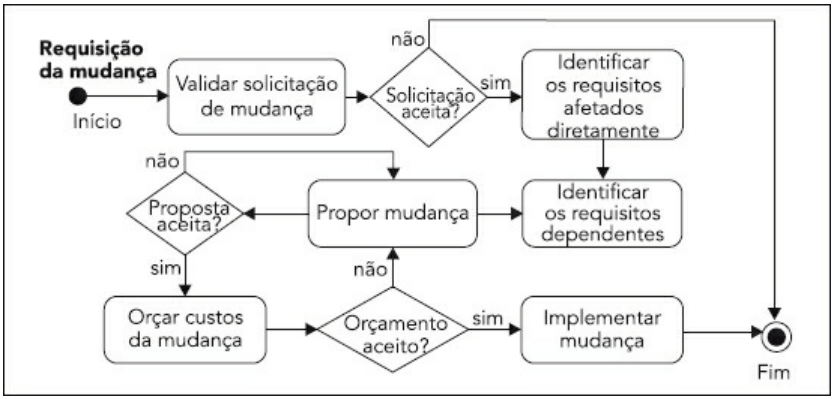

Figura 25 – Gerenciamento de mudanças.
Fonte: Adaptado de Kotonya e Sommerville, 1998.

É importante destacar que toda alteração aprovada gera uma atualização na especificação de requisitos do sistema.

9.2 Identificação de requisitos

Uma precondição para o gerenciamento de requisitos é que cada requisito deve possuir algum tipo de identificação única. A forma mais comum é a numeração sequencial, conforme o capítulo e a sessão em que os requisitos se encontram na documentação.

Além disso, pode-se utilizar a numeração dinâmica, através das referências cruzadas em processadores de textos, o registro dos requisitos em banco de dados ou,

ainda, um esquema de identificação simbólica.

A vantagem da identificação simbólica é que o próprio identificador do requisito, por si só, apresenta informações sobre ele mesmo, conforme os seguintes exemplos:

- RF01: indica que se trata de um requisito funcional;
- RFE02: indica que se trata de um requisito funcional estável;
- RFV03: indica que se trata de um requisito funcional volátil;
- RNF04: indica que se trata de um requisito não funcional.

Assim como nos exemplos, cada organização pode definir um esquema que simbolize as informações importantes para os seus projetos.

9.3 Armazenamento de requisitos

Quanto ao armazenamento dos requisitos, uma das possibilidades é armazenar todos os requisitos em um único documento. Essa opção apresenta como desvantagens a necessidade de manter as informações sobre dependências externamente, além da pesquisa limitada às funcionalidades do processador de textos, dificuldade para o versionamento dos requisitos e de navegação entre requisitos relacionados.

A opção de separação da especificação em diferentes documentos facilita o controle de versionamento, já que são necessárias menos alterações em cada um dos documentos do que se todas as informações estivessem em um único local. Essa opção também facilita a organização e a localização das informações, visto que cada documento tem um objetivo bem específico.

Assim, uma possibilidade é a divisão dos requisitos funcionais, requisitos não funcionais e glossário em documentos distintos. Dependendo da quantidade de requisitos funcionais, pode-se, ainda, subdividi-los em diferentes documentos conforme os módulos do sistema.

Contudo, o armazenamento também é feito em processadores de texto e, consequentemente, possui as mesmas limitações.

Um terceira opção é o armazenamento dos requisitos em banco de dados, o que é possível através do uso de ferramentas para o gerenciamento de requisitos como o Requisite Pro, CaliberRM e DOORS, já vistos no Capítulo 4.

Neste caso, pode-se usufruir de todas as possiblidades oferecidas pelas ferramentas, como controle automatizado de versões, geração de relatórios diversos,

controle de acesso, pesquisa avançada, entre muitas outras.

9.4 Rastreabilidade

Uma parte crítica do processo de gerenciamento de mudanças é a avaliação do impacto de uma mudança no restante do sistema. Se a mudança é proposta na fase de engenharia de requisitos, deve-se avaliar como esta mudança afeta os outros requisitos. Se a mudança é proposta enquanto o sistema está sendo desenvolvido, a avaliação do impacto envolve avaliar como as mudanças afetam os requisitos, o projeto do sistema e sua implementação. Se a mudança é proposta após o sistema entrar em operação, é necessário também avaliar como todos envolvidos no sistema serão afetados pela mudança (KOTONYA e SOMMERVILLE, 1998).

Para realizar este tipo de avaliação de impacto, informações sobre dependências entre requisitos, justificativa e implementação devem ser mantidas para completar a informação na ERS. Isso normalmente é chamado de informação de rastreabilidade. Avaliação do impacto da mudança depende desta informação de rastreabilidade para descobrir quais requisitos serão afetados por uma mudança proposta.

Segundo Gotel (1996), o rastreamento de requisitos é a habilidade para descrever e seguir a vida de um requisito desde sua origem, durante o seu desenvolvimento e nos futuros refinamentos.

Conforme apresentado no quadro 21, existem diferentes tipos de rastreabilidade, os quais podem ser exemplificados mais concretamente por relacionamentos entre informações específicas na ERS e outros documentos do sistema (KOTONYA e SOMMERVILLE, 1998).

Quadro 21 – Tipos de rastreabilidade

Tipo de rastreabilidade	**Descrição**
Requisito – fonte	Relacionamento entre requisitos e documentos ou pessoas que os especificaram.
Requisito – justificativa	Relacionamento entre requisito e a descrição do porquê ele foi especificado.
Requisito – requisito	Relacionamento entre requisito e outros requisitos que são, de alguma maneira, dependentes dele.
Requisito –	Relacionamento entre requisito e subsistemas em que estes requisitos são

arquitetura	implementados.
Requisito – projeto	Relacionamento entre requisito e hardware e componentes de software específicos no sistema que são usados para implementar o requisito.
Requisito – interface	Relacionamento entre requisito e interfaces de sistemas externos que são usados no fornecimento dos requisitos.

Fonte: Kotonya e Sommerville, 1998.

Para facilitar este rastreamento de requisitos, frequentemente são utilizadas tabelas de rastreabilidade, que criam associações entre os requisitos. Uma forma de visualização gráfica desta rastreabilidade é uma matriz de rastreabilidade (Quadro 22), que mostra a ligação entre os requisitos, em que a linha é dependente da coluna e a coluna depende da linha. Esta matriz demonstra de que forma um requisito influencia em outro, possibilitando uma análise do impacto de uma alteração do requisito (KOTONYA e SOMMERVILLE, 1998).

Quadro 22 – Matriz de rastreabilidade de requisitos

	RF1	RF2	RF3	RNF1	RNF2
RF1			X	X	
RF2					X
RF3				X	X
RNF1					X
RNF2				X	

No exemplo (Quadro 22), RF1 é dependente de RF3 e RNF1, R2 é dependente de RNF2, RF3 é dependente de RNF1 e RNF2, RNF1 é dependente de RNF2 e RNF2 é dependente de RNF1. Assim, se for proposta uma alteração no requisito RF3, a leitura da coluna RF3 indica que deve ser avaliado se tal alteração impacta o RF1. Da mesma forma, se for proposta uma alteração no RNF1, a sua respectiva coluna indica que deve ser avaliado um possível impacto no RF1, RF3 e RNF3.

Além das dependências diretas, explicitadas anteriormente, é possível identificar na matriz as dependências indiretas e bidirecionais. No exemplo apresentado no quadro 22, como o RF1 depende diretamente do RNF1 e este depende do RNF2, podemos identificar que RF1 depende, indiretamente, do RNF2. Também é possível identificar um relacionamento bidirecional entre os requisitos RNF1 e RNF2, visto que

tanto o RNF1 depende do RNF2 como o RNF2 depende do RNF1.

Uma outra possibilidade para o registro das dependências entre os requisitos é a tabela de rastreabilidade, a qual apresenta as informações em apenas duas colunas, conforme ilustrado no quadro 23.

Quadro 23 – Tabela de rastreabilidade de requisitos

Requisito	Depende de:	Impacta em:
RF1	RF3 e RNF1	Nenhum
RF2	RNF2	Nenhum
RF3	RNF1 e RNF2	RF1
RNF1	RNF2	RF1, RF3 e RNF2
RNF2	RNF1	RF2, RF3 e RNF1

Além disso, também existe a possiblidade de registro e visualização das dependências entre os requisitos através de ferramentas de apoio, como é o caso do Requisite Pro. As figuras 26 e 27 apresentam as duas possibilidades de definição da rastreabilidade no Requisite Pro, sendo elas a matriz e a árvore de rastreabilidade.

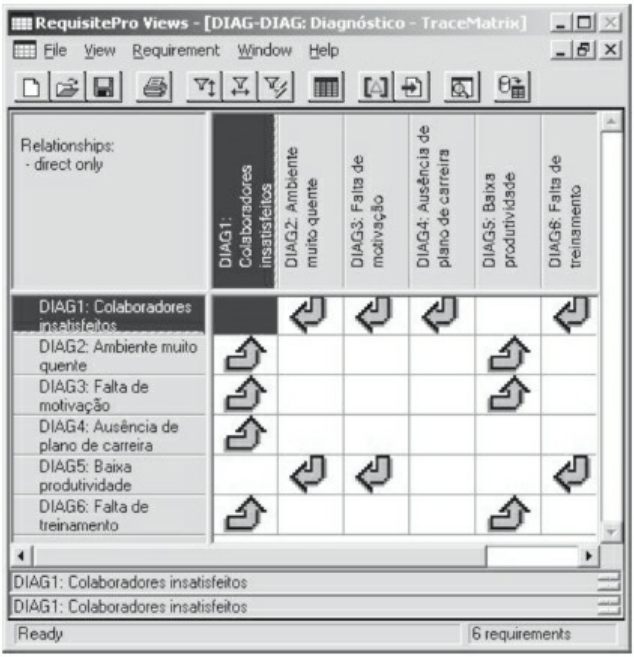

Figura 26 – Matriz de rastreabilidade no IBM RequisitePro.
Fonte: http://www.ibm.com/software/awdtools/reqpro

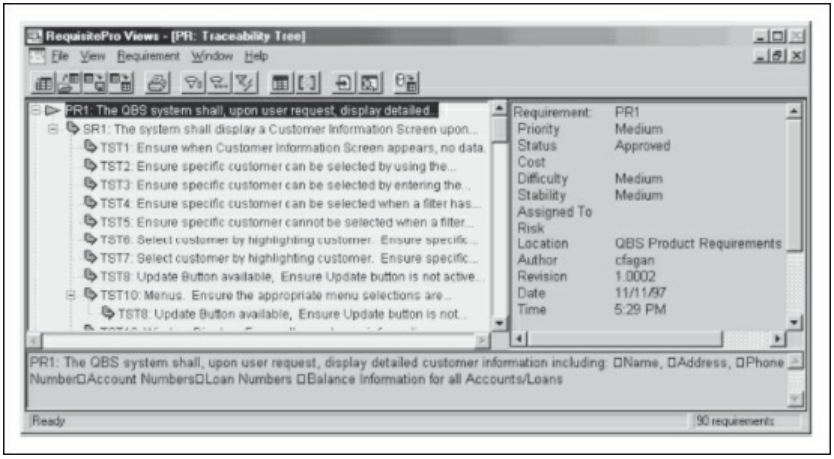

Figura 27 – Árvore de rastreabilidade no RequisitePro.
Fonte: http://www.ibm.com/software/awdtools/reqpro

Apesar dos inúmeros benefícios, a rastreabilidade gera altos custos para coletar, analisar e manter as informações. Por isso, para cada tipo de projeto deve-se definir o tipo de rastreabilidade a ser utilizado, a técnica para identificação, consulta e manutenção (tabela, matriz, banco de dados), o responsável por coletar e manter, a estratégia para mudanças emergenciais e a estratégia para atualização das informações sobre rastreabilidade após uma mudança.

Os principais fatores que influenciam essas definições são o número de requisitos, o tempo de vida do sistema, o nível de maturidade do processo, o tamanho da equipe e o tipo de sistema.

PARA COMPLEMENTAÇÃO DE ESTUDO

Utilize as referências Gotel (1996), Kotonya e Sommervile (1998) e Sommerville (2005) para complementar seu estudo sobre gerenciamento de requisitos.

REFERÊNCIAS BIBLIOGRÁFICAS

BERTINI, Lilian A.; *et. al. Técnicas de Inspeção de Documentos de Requisitos de Software: um Estudo Comparativo*. Workshop de Engenharia de Requisitos – WER 2006.

BOOCH, G. *Oriented-objected Analysis and Design with Applications*. Redwood City, CA: Benjamin Cummings, 1994.

BOOCH, G. *et al. The Unified Modeling Language User Guide*. Reading, MA: Addison Wesley Longman, 1999.

CONSTANTINE, L. L.; YOURDON, E. *Structured Design*. Englewood Cliffs, NJ: Prentice Hall, 1979.

COCKBURN, Alistair. *Escrevendo Casos de Uso Eficazes*. Porto Alegre: Bookman, 2004.

CHRISSIS, M. B.; KONRAD, M.; SHRUM, S. *CMMI: Guidelines for software integration and product improvement*. [S. L.]: Addison Wesley, 2003, 752 p.

EVANS, G. *A simplified approach to RUP*. 2003. Disponível em: <www.ibm.com/developerworks/rational/library/354.html>. Acesso em 22/10/2010.

FOWLER, Martin; *et al. UML Essencial*. Terceira edição. Porto Alegre: Bookman, 2005.

GANE, C.; SARSON, T. *Structured System Analysis*. Englewood Cliffs, NJ: Prentice Hall, 1979.

GOTEL, Orlena; FINKELSTEIN, Anthony. *An Analysis of the Requirements Traceability Problem*. 1996. Disponível em: <www.eprints.ucl.ac.uk/749/1/2.2_rtprob.pdf>. Acesso em outubro de 2010.

HUMPHREY, Watts. *Managing the Software Process*. Reading: Addison-Wesley, 2004.

IEEE. *Recommended Practice for Software Requirements Specifications*. Std 830, 1998.

JACKSON, M. A. *System Development*. Londres: Prentice Hall, 1983.

JACOBSON, I. *et al. Object-oriented software engineering*. Addison Wesley, 1993.

KOTONYA, G.; SOMMERVILLE, I. *Requirements engineering*: processes and techniques. Chichester: John Wiley & Sons, 1998, 282 p.

LARMAN, Craig. *Utilizando UML e Padrões. Uma introdução à análise e ao projeto orientados a objetos e ao desenvolvimento iterativo*. Porto Alegre: Bookman, 2007.

LEFFINGWELL, D.; WIDRIG, D. *Managing Software Requirements: A use case approach*. 2nd. ed. USA: Addison-Wesley, 2003, p.124-125.

OMG. *UML 2.0 Infraestructure Specification*. Object Management Group. 2003. http://www.uml.org

PMI, PROJECT MANAGEMENT INSTITUTE. *A guide to the Project management body of knowledge – PMBOK 3rd edition*. Philadelphia: Project Management Institute, 2004.

PMI, PROJECT MANAGEMENT INSTITUTE. *Um Guia do conhecimento em gerenciamento de projetos – PMBOK*. 4. ed. Philadelphia: Project Management Institute, 2008.

PRESSMAN, Roger S. *Software Engineering:* a practitioner's approach. 5. ed. Boston: McGraw-Hill, 2001, 860 p.

ROBINSON, P. J. *Hierarchical Object-Oriented Design*. Englewood Cliffs, NJ: Prentice Hall, 1992.

RUMBAUGH, J. *et al*. *Object-Oriented Modeling and Design*. Englewood Cliffs, NJ: Prentice Hall, 1991.

SOMMERVILLE, Ian. *Engenharia de Software*. 6. ed. São Paulo: Addison-Wesley, 2005, 592 p.

UNIVERSIDADE DO VALE DO RIO DOS SINOS – UNISINOS

Reitor
Pe. Marcelo Fernandes de Aquino, SJ

Vice-reitor
Pe. José Ivo Follmann, SJ

EDITORA UNISINOS

Diretor
Pe. Pedro Gilberto Gomes, SJ

Editora da Universidade do Vale do Rio dos Sinos
EDITORA UNISINOS
Av. Unisinos, 950
93022-000 São Leopoldo RS Brasil

Telef.: 51.3590 8239
Fax: 51.3590 8238
editora@unisinos.br

© do autor, 2011

2011 Direitos de publicação e comercialização da

Editora da Universidade do Vale do Rio dos Sinos
EDITORA UNISINOS

S729e Souza, Vinícius Costa de.
Engenharia de requisitos de software / Vinícius Costa de Souza. – São Leopoldo: Unisinos, 2011.

86 p.: il. – (Coleção EaDUnisinos).

Inclui bibliografia.
ISBN 978-85-7431-431-0

1. Software – Desenvolvimento. I. Título. II. Série.
CDU 004.414.38

Dados Internacionais de Catalogação na Publicação (CIP)
(Bibliotecária Fabiane Pacheco Martino - CRB 10/1256)

Esta obra segue as normas do Acordo Ortográfico da Língua Portuguesa vigente desde 2009.

COLEÇÃO EAD

Editor
Carlos Alberto Gianotti

Acompanhamento editorial
Mateus Colombo Mendes

Revisão	*Editoração*	*Capa*
Renato Deitos	Rafael Tarcísio Forneck	Isabel Carballo

Impressão, verão de 2011

A reprodução, ainda que parcial, por qualquer meio, das páginas que compõem este livro, para uso não individual, mesmo para fins didáticos, sem autorização escrita do editor, é ilícita e constitui uma contrafação danosa à cultura.
Foi feito o depósito legal.

Sobre o autor

VINÍCIUS COSTA DE SOUZA é mestre em Computação Aplicada pela Universidade do Vale do Rio

dos Sinos – UNISINOS (2005) e bacharel em Informática – habilitação em Análise de Sistemas pela UNISINOS (2002). Atua no ensino superior desde 2004 e possui experiência de dez anos no desenvolvimento de sistemas, principalmente baseados na Web. Ministra disciplinas da área de Engenharia de Software nos cursos de graduação em Ciência da Computação e Sistemas de Informação e atua como coordenador executivo e professor da Graduação Tecnológica em Análise e Desenvolvimento de Sistemas e da Especialização em Qualidade de Software da UNISINOS.

Edição digital: dezembro 2013

Arquivo ePub produzido pela **Simplíssimo Livros**