Capítulo 1 – Introdução

Sistemas de software são reconhecidamente importantes ativos estratégicos para diversas organizações. Uma vez que tais sistemas, em especial os sistemas de informação, têm um papel vital no apoio aos processos de negócio das organizações, é fundamental que os sistemas funcionem de acordo com os requisitos estabelecidos. Neste contexto, uma importante tarefa no desenvolvimento de software é a identificação e o entendimento dos requisitos dos negócios que os sistemas vão apoiar (AURUM; WOHLIN, 2005).

A Engenharia de Requisitos é o processo pelo qual os requisitos de um produto de software são coletados, analisados, documentados e gerenciados ao longo de todo o ciclo de vida do software (AURUM; WOHLIN, 2005). Este texto aborda o processo de Engenharia de Requisitos, concentrando-se nas atividades de levantamento de requisitos e modelagem conceitual.

Este capítulo apresenta o tema e como o mesmo é tratado neste texto. A Seção 1.1 discute a relação entre a Engenharia de Requisitos e o processo de software. A Seção 1.2 apresenta a organização deste texto.

1.1 – Desenvolvimento de Software e Engenharia de Requisitos

Um processo de software envolve diversas atividades que podem ser classificadas quanto ao seu propósito em:

- Atividades de Desenvolvimento (ou Técnicas): são as atividades diretamente relacionadas ao processo de desenvolvimento do software, ou seja, que contribuem diretamente para o desenvolvimento do produto de software a ser entregue ao cliente. São exemplos de atividades de desenvolvimento: levantamento e análise de requisitos, projeto e implementação.
- Atividades de Gerência: envolvem atividades relacionadas ao gerenciamento do projeto de maneira abrangente. Incluem, dentre outras: atividades de planejamento e acompanhamento gerencial do projeto (processo de Gerência de Projetos), tais como realização de estimativas, elaboração de cronogramas, análise dos riscos do projeto etc.; atividades relacionadas à gerência da evolução dos diversos artefatos produzidos nos projetos de software (processo de Gerência de Configuração); atividades relacionadas à gerência de ativos reutilizáveis de uma organização (processo de Gerência de Reutilização) etc.
- Atividades de Controle da Qualidade: são aquelas relacionadas com a avaliação da qualidade do produto em desenvolvimento e do processo de software utilizado. Incluem atividades de verificação, validação e garantia da qualidade.

As atividades de desenvolvimento formam a espinha dorsal do desenvolvimento e são realizadas segundo uma ordem estabelecida no planejamento. As atividades de gerência e de

controle da qualidade são, muitas vezes, ditas atividades de apoio, pois não estão ligadas diretamente à construção do produto final, ou seja, o software a ser entregue para o cliente, incluindo toda a documentação necessária. Essas atividades, normalmente, são realizadas ao longo de todo o ciclo de vida, sempre que necessário ou em pontos pré-estabelecidos durante o planejamento, ditos marcos ou pontos de controle. A Figura 1.1 mostra a relação entre esses tipos de atividades.

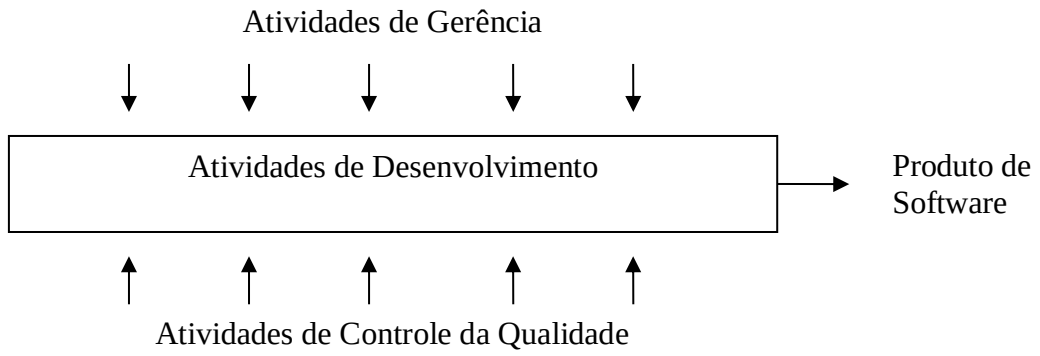

Figura 1.1 – Atividades do Processo de Software

No que concerne às atividades técnicas, tipicamente o processo de software inicia-se com o Levantamento de Requisitos, quando os requisitos do sistema a ser desenvolvido são preliminarmente capturados e organizados. Uma vez capturados, os requisitos devem ser modelados, avaliados e documentados. Uma parte essencial dessa fase é a elaboração de modelos descrevendo *o quê* o software tem de fazer (e não *como* fazê-lo), dita Modelagem Conceitual. Até este momento, a ênfase está sobre o domínio do problema e não se deve pensar na solução técnica, computacional a ser adotada.

Com os requisitos pelo menos parcialmente capturados e especificados na forma de modelos, pode-se começar a trabalhar no domínio da solução. Muitas soluções são possíveis para o mesmo conjunto de requisitos e elas são intrinsecamente ligadas a uma dada plataforma de implementação (linguagem de programação, mecanismo de persistência a ser adotado etc.). A fase de projeto tem por objetivo definir e especificar uma solução a ser implementada. É uma fase de tomada de decisão, tendo em vista que muitas soluções são possíveis.

Uma vez projetado o sistema, pode dar-se início à implementação, quando as unidades de software do projeto são implementadas e testadas individualmente. Gradativamente, os elementos vão sendo integrados e testados (teste de integração), até se obter o sistema, quando o todo deve ser testado (teste de sistema). Por fim, uma vez testado no ambiente de desenvolvimento, o software pode ser colocado em produção. Usuários devem ser treinados, o ambiente de produção deve ser configurado e o sistema deve ser instalado e testado, agora pelos usuários no ambiente de produção (testes de homologação ou aceitação). Caso o software demonstre prover as capacidades requeridas, ele pode ser aceito e a operação iniciada.

Requisitos têm um papel central no desenvolvimento de software, uma vez que uma a principal medida do sucesso de um software é o grau no qual ele atende aos objetivos e requisitos para os quais foi construído. Requisitos são a base para estimativas, modelagem, projeto, implementação, testes e até mesmo para a manutenção. Portanto, estão presentes ao longo de todo o ciclo de vida de um software.

Nos estágios iniciais de um projeto, requisitos têm de ser levantados, entendidos e documentados (atividades de Levantamento, Análise e Documentação de Requisitos). Dada a

importância dos requisitos para o sucesso de um projeto, atividades de controle da qualidade devem ser realizadas para verificar, validar e garantir a qualidade dos requisitos, uma vez que os custos serão bem maiores se defeitos em requisitos forem identificados tardiamente. Mesmo quando coletados de forma sistemática, requisitos mudam. Os negócios são dinâmicos e não há como garantir que os requisitos não sofrerão alterações. Assim, é fundamental gerenciar a evolução dos requisitos, bem como manter a rastreabilidade entre os requisitos e os demais artefatos produzidos no projeto (atividade de Gerência de Requisitos).

Como se pode observar, o tratamento de requisitos envolve atividades de desenvolvimento (Levantamento, Análise e Documentação de Requisitos), gerência (Gerência de Requisitos) e controle da qualidade (Verificação, Validação e Garantia da Qualidade de Requisitos). Ao conjunto de atividades relacionadas a requisitos, dá-se o nome de Processo de Engenharia de Requisitos.

1.2 - A Organização deste Texto

Este texto procura oferecer uma visão geral da Engenharia de Requisitos de Software, discutindo as principais atividades desse processo e como realizá-las. Ênfase especial é dada à aplicação das técnicas de modelagem e especificação de Sistemas de Informação, i.e., sistemas desenvolvidos para apoiar processos de negócio das organizações.

Nos capítulos que se seguem, os seguintes temas são abordados:

- Capítulo 2 – *Engenharia de Requisitos de Software*: inicia discutindo o que são requisitos e tipos e níveis de requisitos. A seguir, apresenta o processo de engenharia de requisitos considerado neste texto, o qual contém as seguintes atividades: Levantamento de Requisitos, Análise de Requisitos, Documentação de Requisitos, Verificação e Validação de Requisitos e Gerência de Requisitos. Discute-se, também, como as principais normas e modelos de qualidade de processos de software tratam a questão dos requisitos.
- Capítulo 3 – *Levantamento de Requisitos*: provê uma visão geral do processo de levantamento de requisitos e aborda técnicas para levantar requisitos, dentre elas entrevistas, questionários, observação, investigação de documentos e prototipagem. Aborda-se, também, a modelagem de processos de negócio e como ela pode apoiar o levantamento de requisitos. Finalmente, discute-se como escrever e documentar requisitos de cliente.
- Capítulo 4 – *Análise de Requisitos*: trata da análise de requisitos, discutindo a análise e especificação de requisitos funcionais (modelagem conceitual) e não funcionais. O capítulo inicia provendo uma introdução à modelagem conceitual. Na sequência é apresentada brevemente a Linguagem de Modelagem Unificada (*Unified Modeling Language* – UML), amplamente usada na Análise de Requisitos e os conceitos da orientação a objetos, paradigma adotado neste texto. Um método de análise de requisitos funcionais é apresentado. A seguir, discute-se a especificação de requisitos não funcionais. Por fim, a documentação da atividade de análise de requisitos é abordada.
- Capítulo 5 – *Modelagem de Objetivos e de Casos de Uso*: discute a análise de objetivos como uma ferramenta para apoiar a identificação de requisitos e as razões que os justificam, e o papel da modelagem de casos de uso na especificação de

requisitos funcionais de sistema. Apresenta os elementos centrais da modelagem de casos de uso, o diagrama de casos de uso da UML e discute como descrever casos de uso textualmente.

- Capítulo 6 – *Modelagem Estrutural*: trata da modelagem dos principais conceitos do domínio, suas relações e propriedades, permitindo representar o conhecimento do domínio relevante para o sistema em desenvolvimento. A elaboração de diagramas de classes da UML é o foco deste capítulo.
- Capítulo 7 – *Modelagem Dinâmica*: aborda os modelos usados para especificar as mudanças válidas no estado dos objetos de domínio, bem como para modelar o comportamento esperado do sistema. O foco deste capítulo é a elaboração de diagramas de estados e diagramas de atividades.
- Capítulo 8 – *Qualidade e Agilidade em Requisitos*: explora técnicas de leitura de modelos, visando apoiar a verificação da consistência entre os diversos artefatos produzidos durante a análise de requisitos; apresenta o enfoque da modelagem ágil; e discute abordagens para reutilização na Engenharia de Requisitos, dentre elas Engenharia de Domínio, Ontologias e Padrões de Análise.

Referências do Capítulo

AURUM, A., WOHLIN, C., *Engineering and Managing Software Requirements*, Springer-Verlag, 2005.

Capítulo 2 – Engenharia de Requisitos de Software

Requisitos têm um papel central no processo de software, sendo considerados um fator determinante para o sucesso ou fracasso de um projeto de software. O processo de levantar, analisar, documentar, gerenciar e controlar a qualidade dos requisitos é chamado de Engenharia de Requisitos.

Este capítulo dá uma visão geral da Engenharia de Requisitos. A Seção 2.1 apresenta diferentes definições do conceito de requisito e trata de tipos e níveis de requisitos. A Seção 2.2 aborda o processo de engenharia de requisitos, discutindo cada uma de suas atividades. Finalmente, a Seção 2.3 discute como as principais normas e modelos de qualidade de processos de software tratam a questão dos requisitos.

2.1 – Requisitos

Existem diversas definições para requisito de software na literatura, dentre elas:

- Requisitos são descrições dos serviços que devem ser providos pelo sistema e de suas restrições operacionais (SOMMERVILLE, 2007).
- Um requisito é uma característica do sistema ou a descrição de algo que o sistema é capaz de realizar para atingir seus objetivos (PFLEEGER, 2004).
- Um requisito é alguma coisa que o produto tem de fazer ou uma qualidade que ele precisa apresentar (ROBERTSON; ROBERTSON, 2006).

Com base nessas e em outras definições, pode-se dizer que os requisitos de um sistema incluem especificações dos serviços que o sistema deve prover, restrições sob as quais ele deve operar, propriedades gerais do sistema e restrições que devem ser satisfeitas no seu processo de desenvolvimento.

As várias definições acima apresentadas apontam para a existência de diferentes tipos de requisitos. Uma classificação amplamente aceita quanto ao tipo de informação documentada por um requisito faz a distinção entre requisitos funcionais e requisitos não funcionais.

- **Requisitos Funcionais**: são declarações de serviços que o sistema deve prover, descrevendo o que o sistema deve fazer (SOMMERVILLE, 2007). Um requisito funcional descreve uma interação entre o sistema e o seu ambiente (PFLEEGER, 2004), podendo descrever, ainda, como o sistema deve reagir a entradas específicas, como o sistema deve se comportar em situações específicas e o que o sistema não deve fazer (SOMMERVILLE, 2007).
- **Requisitos Não Funcionais**: descrevem restrições sobre os serviços ou funções oferecidos pelo sistema (SOMMERVILLE, 2007), as quais limitam as opções para criar uma solução para o problema (PFLEEGER, 2004). Neste sentido, os requisitos não funcionais são muito importantes para a fase de projeto (design), servindo como base para a tomada de decisões nessa fase.

Os requisitos não funcionais têm origem nas necessidades dos usuários, em restrições de orçamento, em políticas organizacionais, em necessidades de interoperabilidade com outros sistemas de software ou hardware ou em fatores externos como regulamentos e legislações (SOMMERVILLE, 2007). Assim, os requisitos não funcionais podem ser classificados quanto à sua origem. Existem diversas classificações de requisitos não funcionais. Sommerville (2007), por exemplo, classifica-os em:

- **Requisitos de produto**: especificam o comportamento do produto (sistema). Referem-se a atributos de qualidade que o sistema deve apresentar, tais como confiabilidade, usabilidade, eficiência, portabilidade, manutenibilidade e segurança.
- **Requisitos organizacionais**: são derivados de metas, políticas e procedimentos das organizações do cliente e do desenvolvedor. Incluem requisitos de processo (padrões de processo e modelos de documentos que devem ser usados), requisitos de implementação (tal como a linguagem de programação a ser adotada), restrições de entrega (tempo para chegar ao mercado - *time to market*, restrições de cronograma etc.), restrições orçamentárias (custo, custo-benefício) etc.
- **Requisitos externos**: referem-se a todos os requisitos derivados de fatores externos ao sistema e seu processo de desenvolvimento. Podem incluir requisitos de interoperabilidade com sistemas de outras organizações, requisitos legais (tais como requisitos de privacidade) e requisitos éticos.

No que se refere aos RNFs de produto, eles podem estar relacionados a propriedades emergentes do sistema como um todo, ou seja, propriedades que não podem ser atribuídas a uma parte específica do sistema, mas que, ao contrário, só aparecem após a integração de seus componentes, tal como confiabilidade (SOMMERVILLE, 2007). Contudo, algumas vezes, essas características podem estar associadas a uma função específica ou a um conjunto de funções. Por exemplo, uma certa função pode ter restrições severas de desempenho, enquanto outras funções do mesmo sistema podem não apresentar tal restrição.

Os requisitos devem ser redigidos de modo a serem passíveis de entendimento pelos diversos interessados (*stakeholders*). Clientes[1], usuários finais e desenvolvedores são todos interessados em requisitos, mas têm expectativas diferentes. Enquanto desenvolvedores e usuários finais têm interesse em detalhes técnicos, clientes requerem descrições mais abstratas. Assim, é útil apresentar requisitos em diferentes níveis de descrição. Sommerville (2007) sugere dois níveis de descrição de requisitos:

- **Requisitos de Cliente ou de Usuário:** são declarações em linguagem natural acompanhadas de diagramas intuitivos de quais serviços são esperados do sistema e das restrições sob as quais ele deve operar. Devem estar em um nível de abstração mais alto, de modo que sejam compreensíveis pelos clientes e usuários do sistema que não possuem conhecimento técnico.

[1] É importante notar a distinção que se faz aqui entre clientes e usuários finais. Consideram-se clientes aqueles que contratam o desenvolvimento do sistema e que, muitas vezes, não usarão diretamente o sistema. Eles estão mais interessados nos resultados da utilização do sistema pelos usuários do que no sistema em si. Usuários, por outro lado, são as pessoas que utilizarão o sistema em seu dia a dia. Ou seja, os usuários são as pessoas que vão operar ou interagir diretamente com o sistema.

- **Requisitos de Sistema:** definem detalhadamente as funções, serviços e restrições do sistema. São versões expandidas dos requisitos de cliente usados pelos desenvolvedores para projetar, implementar e testar o sistema. Como requisitos de sistema são mais detalhados, as especificações em linguagem natural são insuficientes e para especificá-los, notações mais especializadas devem ser utilizadas.

Vale destacar que esses níveis de descrição de requisitos são aplicados em momentos diferentes e com propósitos distintos. Requisitos de cliente são elaborados nos estágios iniciais do desenvolvimento (levantamento preliminar de requisitos) e servem de base para um entendimento entre clientes e desenvolvedores acerca do que o sistema deve contemplar. Esses requisitos são, normalmente, usados como base para a contratação e o planejamento do projeto. Requisitos de sistema, por sua vez, são elaborados como parte dos esforços diretos para o desenvolvimento do sistema, capturando detalhes importantes para as fases técnicas posteriores do processo de desenvolvimento, a saber: projeto, implementação e testes.

Entretanto, não se deve perder de vista que requisitos de sistema são derivados dos requisitos de cliente. Os requisitos de sistema acrescentam detalhes, explicando os serviços e funções a serem providos pelo sistema em desenvolvimento. Os interessados nos requisitos de sistema necessitam conhecer mais precisamente o que o sistema fará, pois eles estão preocupados com o modo como o sistema apoiará os processos de negócio ou porque estão envolvidos na sua construção (SOMMERVILLE, 2007).

Uma vez que requisitos de cliente e de sistema têm propósitos e público alvo diferentes, é útil descrevê-los em documentos diferentes. Pfleeger (2004) sugere que dois tipos de documentos de requisitos sejam elaborados:

- **Documento de Definição de Requisitos**: deve ser escrito de maneira que o cliente possa entender, i.e., na forma de uma listagem do quê o cliente espera que o sistema proposto faça. Ele representa um consenso entre o cliente e o desenvolvedor sobre o quê o cliente quer.
- **Documento de Especificação de Requisitos**: refina os requisitos de cliente em termos mais técnicos, apropriados para o desenvolvimento de software, sendo produzido por analistas de requisitos.

Vale ressaltar que deve haver uma correspondência direta entre cada requisito de usuário listado no documento de requisitos e os requisitos de sistema tratados no documento de especificação de requisitos.

2.2 – O Processo de Engenharia de Requisitos

A Engenharia de Requisitos (ER) é o ramo da Engenharia de Software que envolve as atividades relacionadas com a definição dos requisitos de software de um sistema, desenvolvidas ao longo do ciclo de vida de software (KOTONYA; SOMMERVILLE, 1998). O processo de ER envolve criatividade, interação entre pessoas, conhecimento e experiência para transformar informações diversas (sobre a organização, sobre leis, sobre o sistema a ser construído etc.) em documentos e modelos que direcionem o desenvolvimento de software (KOTONYA; SOMMERVILLE, 1998).

A Engenharia de Requisitos é fundamental, pois possibilita, dentre outros, estimar custo e tempo de maneira mais precisas e melhor gerenciar mudanças em requisitos. Dentre os problemas de um processo de engenharia de requisitos ineficiente, podem-se citar (KOTONYA; SOMMERVILLE, 1998): (i) requisitos inconsistentes, (ii) produto final com custo maior do que o esperado, (iii) software instável e com altos custos de manutenção e (iv) clientes insatisfeitos.

A Engenharia de Requisitos pode ser descrita como um processo, ou seja, um conjunto organizado de atividades que deve ser seguido para derivar, avaliar e manter os requisitos e artefatos relacionados. Uma descrição de um processo, de forma geral, deve incluir, além das atividades a serem seguidas, a estrutura ou sequência dessas atividades, quem é responsável por cada atividade, suas entradas e saídas, as ferramentas usadas para apoiar as atividades e os métodos, técnicas e diretrizes a serem seguidos na sua realização.

Processos de ER podem variar muito de uma organização para outra, ou até mesmo dentro de uma organização específica, em função de características dos projetos. A definição de um processo apropriado à organização traz muitos benefícios, pois uma boa descrição do mesmo fornecerá orientações e reduzirá a probabilidade de esquecimento ou de uma execução superficial. No entanto, não faz sentido falar em processo ideal ou definir algum e impô-lo a uma organização. Ao invés disto, as organizações devem iniciar com um processo genérico e adaptá-lo para um processo mais detalhado, que seja apropriado às suas reais necessidades (SOMMERVILLE; SAWYER, 1997). Assim, a implantação de processos em uma organização deve ser feita segundo as necessidades da mesma, ou seja, o processo deve ser definido de acordo com as características da organização. Existem, portanto, fatores que contribuem para a variabilidade do processo de ER, dentre eles a maturidade técnica, o envolvimento disciplinar, a cultura organizacional e os domínios de aplicação nos quais a organização atua (KOTONYA; SOMMERVILLE, 1998).

Wiegers (2003) destaca alguns benefícios que um processo de ER de qualidade pode trazer, dentre elas: menor quantidade de defeitos nos requisitos, redução de retrabalho, desenvolvimento de menos características desnecessárias, diminuição de custos, desenvolvimento mais rápido, menos problemas de comunicação, alterações de escopo reduzidas, estimativas mais confiáveis e maior satisfação de clientes e desenvolvedores.

Ainda que diferentes projetos requeiram processos com características específicas para contemplar suas peculiaridades, é possível estabelecer um conjunto de atividades básicas que deve ser considerado na definição de um processo de ER. Tomando por base o processo proposto por Kotonya e Sommerville (1998), neste texto considera-se que um processo de ER deve contemplar, tipicamente, as atividades mostradas na Figura 2.1: levantamento de requisitos, análise de requisitos, documentação de requisitos, verificação e validação de requisitos e gerência de requisitos.

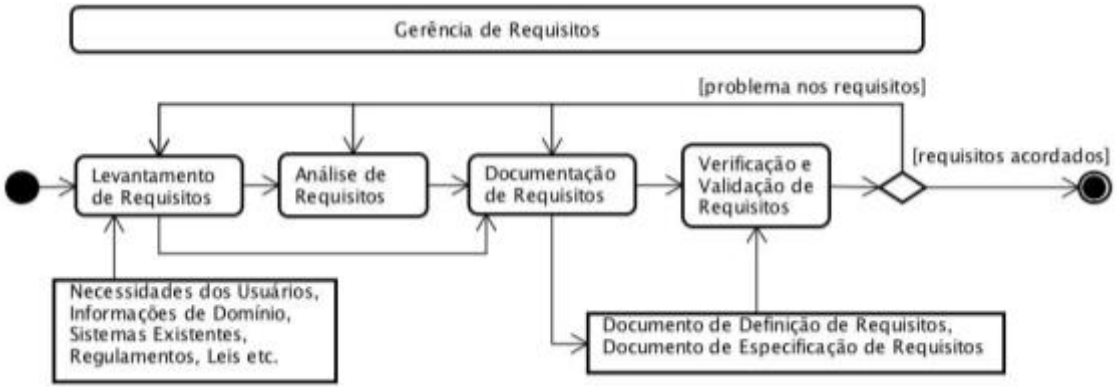

Figura 2.1 – Processo de Engenharia de Requisitos (adaptado de (KOTONYA; SOMMERVILLE, 1998))

O processo começa pelo levantamento de requisitos, que deve levar em conta necessidades dos usuários e clientes, informações de domínio, sistemas existentes, regulamentos, leis etc. Uma vez identificados requisitos, é possível iniciar a atividade de análise, quando os requisitos levantados são usados como base para a modelagem do sistema. Tanto no levantamento quanto na análise de requisitos, é importante documentar requisitos e modelos. Conforme discutido anteriormente, para documentar requisitos, dois documentos são normalmente utilizados: o Documento de Definição de Requisitos, contendo uma lista dos requisitos de cliente identificados, e o Documento de Especificação de Requisitos, que registra os requisitos de sistema e os vários diagramas resultantes do trabalho de análise. Os documentos produzidos são, então, verificados e validados. Adicionalmente, um esforço de garantia da qualidade deve ser realizado, visando garantir conformidade em relação a padrões e ao processo estabelecidos pela organização. Caso clientes, usuários e desenvolvedores estejam de acordo com os requisitos, o processo de desenvolvimento pode avançar; caso contrário, deve-se retornar à atividade correspondente para resolver os problemas identificados. Em paralelo a todas as atividades anteriormente mencionadas, há a gerência de requisitos, que se ocupa em gerenciar mudanças nos requisitos.

Vale destacar que não há limites bem definidos entre as atividades acima citadas. Na prática, elas são intercaladas e existe um alto grau de iteração e feedback entre elas. Idealmente, deve-se começar com um levantamento preliminar de requisitos que considere apenas requisitos de cliente. Esses requisitos devem ser documentados em um Documento de Definição de Requisitos e avaliados (verificação e validação). Esse documento deve ser usado como base para a contratação do projeto. Caso haja acordo em relação aos requisitos de cliente, pode-se iniciar um ciclo de levantamento detalhado de requisitos e análise. O processo é executado até que todos os usuários estejam satisfeitos e concordem com os requisitos ou até que a pressão do cronograma precipite o início da fase de projeto, o que é indesejável (KOTONYA; SOMMERVILLE, 1998).

Além disso, ao se adotar um modelo de ciclo de vida iterativo, essas atividades podem ser realizadas muitas vezes. Neste caso, uma vez contratado o projeto e, portanto, obtido um acordo em relação aos requisitos de cliente iniciais, pode-se iniciar um ciclo de levantamento detalhado de requisitos e análise, produzindo uma versão do Documento de Especificação de Requisitos. Havendo acordo em relação aos requisitos e modelos contidos nessa primeira versão do documento, o desenvolvimento pode prosseguir para a

porção tratada nessa iteração (projeto, implementação e testes), enquanto um novo ciclo de levantamento e análise se inicia para tratar outros requisitos de cliente ainda não contemplados.

A seguir, as atividades do processo de ER proposto são discutidas com um pouco mais de detalhes.

2.2.1 – Levantamento de Requisitos

O levantamento de requisitos corresponde à fase inicial do processo de ER e envolve atividades de descoberta dos requisitos. Nessa fase, um esforço conjunto de clientes, usuários e especialistas de domínio é necessário, com o objetivo de entender a organização, seus processos, necessidades, deficiências dos sistemas de software atuais, possibilidades de melhorias, bem como restrições existentes. Trata-se de uma atividade complexa que não se resume somente a perguntar às pessoas o que elas desejam, mas sim analisar cuidadosamente a organização, o domínio da aplicação e os processos de negócio no qual o sistema será utilizado (KOTONYA; SOMMERVILLE, 1998).

Para levantar quais são os requisitos de um sistema, devem-se obter informações dos interessados (*stakeholders*), consultar documentos, obter conhecimentos do domínio e estudar o negócio da organização. Neste contexto, quatro dimensões devem ser consideradas, como ilustra a Figura 2.2 (KOTONYA; SOMMERVILLE, 1998):

Figura 2.2 - Dimensões do levantamento de requisitos

- **Entendimento do domínio da aplicação:** entendimento geral da área na qual o software a ser desenvolvido está inserido;
- **Entendimento do problema:** entendimento dos detalhes do problema específico a ser resolvido com o auxílio do sistema a ser desenvolvido;
- **Entendimento do negócio:** entender como o sistema afetará a organização e como contribuirá para que os objetivos do negócio e os objetivos gerais da organização sejam atingidos;

- **Entendimento das necessidades e das restrições dos interessados:** entender as demandas de apoio para a realização do trabalho de cada um dos interessados no sistema, entender os processos de trabalho a serem apoiados pelo sistema e o papel de eventuais sistemas existentes na execução e condução dos processos de trabalho. Consideram-se interessados no sistema, todas as pessoas que são afetadas pelo sistema de alguma maneira, dentre elas clientes, usuários finais e gerentes de departamentos onde o sistema será instalado.

A atividade de levantamento de requisitos é dominada por fatores humanos, sociais e organizacionais e envolve pessoas com diferentes conhecimentos e objetivos, o que a torna complexa. Christel e Kang (apud PRESSMAN, 2006) citam alguns problemas que tornam o levantamento de requisitos uma tarefa difícil:

- Problemas de escopo: as fronteiras do sistema são mal definidas ou os clientes/usuários especificam detalhes técnicos desnecessários que podem confundir, em vez de esclarecer, os objetivos globais do sistema.
- Problemas de entendimento: Os clientes/usuários não estão completamente certos do que é necessário, têm pouca compreensão das capacidades e limitações de um sistema computacional, não têm pleno entendimento do domínio do problema, têm dificuldade de comunicar suas necessidades, omitem informação que acreditam ser óbvia, especificam requisitos que conflitam com as necessidades de outros clientes/usuários ou especificam requisitos que são ambíguos ou impossíveis de testar.
- Problemas de volatilidade: Os requisitos mudam ao longo do tempo.

Kotonya e Sommerville (1998) destacam outras dificuldades que complementam e reforçam os problemas apontados por Christel e Kang, a saber:

- Pode ser difícil compreender e coletar informações quando existem muitos termos desconhecidos, manuais técnicos etc.
- Pessoas que entendem o problema a ser resolvido podem ser muito ocupadas e não ter muito tempo para, juntamente como analista, levantar os requisitos e entender o sistema.
- Políticas organizacionais podem influenciar nos requisitos de um sistema.
- Os interessados não sabem muito bem o que querem do sistema e não conhecem muitos termos.

Diversas técnicas podem ser utilizadas no levantamento de requisitos, as quais podem possuir diferentes objetos de investigação ou podem ter foco em tipos diferentes de requisitos. Assim, é útil empregar várias dessas técnicas concomitantemente, de modo a se ter um levantamento de requisitos mais eficaz. Dentre as várias técnicas, podem ser citadas (KENDALL; KENDALL, 2010; KOTONYA; SOMMERVILLE, 1998; AURUM; WOHLIN, 2005):

- Entrevistas: técnica amplamente utilizada, que consiste em conversas direcionadas com um propósito específico e com formato “pergunta-resposta”. Seu objetivo é descobrir problemas a serem tratados, levantar procedimentos importantes e saber a opinião e as expectativas do entrevistado sobre o sistema.

- Questionários: o uso de questionários possibilita ao analista obter informações como postura, crenças, comportamentos e características de várias pessoas que serão afetas pelo sistema.
- Observação: consiste em observar o comportamento e o ambiente dos indivíduos de vários níveis organizacionais. Utilizando-se essa técnica, é possível capturar o que realmente é feito e qual tipo de suporte computacional é realmente necessário. Ajuda a confirmar ou refutar informações obtidas com outras técnicas e ajuda a identificar tarefas que podem ser automatizadas e que não foram identificadas pelos interessados.
- Análise de documentos: pela análise de documentos existentes na organização, analistas capturam informações e detalhes difíceis de conseguir por entrevista e observação. Documentos revelam um histórico da organização e sua direção.
- Cenários: com o uso desta técnica, um cenário de interação entre o usuário final e o sistema é montado e o usuário simula sua interação com o sistema nesse cenário, explicando ao analista o que ele está fazendo e de que informações ele precisa para realizar a tarefa descrita no cenário. O uso de cenários ajuda a entender requisitos, a expor o leque de possíveis interações e a revelar facilidades requeridas.
- Prototipagem: um protótipo é uma versão preliminar do sistema, muitas vezes não operacional e descartável, que é apresentada ao usuário para capturar informações específicas sobre seus requisitos de informação, observar reações iniciais e obter sugestões, inovações e informações para estabelecer prioridades e redirecionar planos.
- Dinâmicas de Grupo: há várias técnicas de levantamento de requisitos que procuram explorar dinâmicas de grupo para a descoberta e o desenvolvimento de requisitos, tais como *Brainstorming* e JAD (*Joint Application Development*). Na primeira, representantes de diferentes grupos de interessados engajam-se em uma discussão informal para rapidamente gerarem o maior número possível de ideias. Na segunda, interessados e analistas se reúnem para discutir problemas a serem solucionados e soluções possíveis. Com as diversas partes envolvidas representadas, decisões podem ser tomadas e questões podem ser resolvidas mais rapidamente. A principal diferença entre JAD e *Brainstorming* é que, em JAD, tipicamente os objetivos do sistema já foram estabelecidos antes dos interessados participarem. Além disso, sessões JAD são normalmente bem estruturadas, com passos, ações e papéis de participantes definidos.

2.2.2 – Análise de Requisitos

Uma vez preliminarmente identificados os requisitos, é possível iniciar a atividade de análise, quando os requisitos levantados devem ser refinados. Neste contexto, modelos são essenciais e diversos tipos de modelos podem ser utilizados. Esses modelos são representações gráficas que descrevem objetivos e processos de negócio, o problema a ser resolvido e o sistema a ser desenvolvido.

De maneira simples, um modelo é uma simplificação da realidade enfocando certos aspectos considerados relevantes segundo a perspectiva do modelo, e omitindo os

demais. Modelos são construídos para se obter uma melhor compreensão da porção da realidade sendo modelada.

Modelos são úteis para a Engenharia de Requisitos (e, de maneira mais geral, para a Engenharia de Software) por vários motivos, dentre eles (LAMSWEERDE, 2009) (PRESSMAN, 2006):

- Modelos permitem enfocar os aspectos chave, em detrimento de detalhes irrelevantes.
- Ajudam o engenheiro de requisitos a entender a informação, a função e o comportamento do sistema, tornando a tarefa de análise de requisitos mais fácil e sistemática.
- Proveem apoio para o entendimento e explanação para os envolvidos.
- Tornam-se o ponto focal para a revisão e, portanto, a chave para a determinação da consistência e correção da especificação. Deste modo, apoiam a análise, detecção e reparo de erros de forma precoce.
- Servem de base para a tomada de decisão.
- Fornecem uma estrutura para as atividades da ER, provendo um alvo para o que deve ser obtido, avaliado, especificado, consolidado e modificado, sendo a base para a geração do Documento de Especificação de Requisitos.

Em essência, a fase de análise é uma atividade de modelagem. A modelagem nesta fase é dita conceitual, pois ela se preocupa com o domínio do problema e não com soluções técnicas para o mesmo. Os modelos de análise são elaborados para se obter uma compreensão maior acerca do sistema a ser desenvolvido e para especificá-lo. Diferentes modelos podem ser construídos para representar diferentes perspectivas. Classicamente, há duas principais perspectivas que são consideradas na fase de análise:

- Perspectiva estrutural: busca modelar os conceitos, propriedades e relações do domínio que são relevantes para o sistema em desenvolvimento. Provê uma visão estática das informações que o sistema necessita tratar e, portanto, refere-se às representações que o sistema terá de prover para abstrair entidades do mundo real. Em outras palavras, esta perspectiva foca em "sobre o quê?" (quais informações?) o sistema opera. Diagramas de classes e modelos de entidades e relacionamentos são usados para modelar esta perspectiva.
- Perspectiva comportamental: visa modelar o comportamento geral do sistema, de suas funcionalidades ou de uma entidade específica ao longo do tempo. Provê uma visão do comportamento do sistema ou de uma parcela do sistema. De maneira bem simples, o foco dessa perspectiva é no "quê" o sistema deve fazer (que funções ou serviços ele deve prover e como ele deve se comportar). Diagramas de casos de uso, diagramas de atividades, diagramas de estados e diagramas de interação são usados para modelar essa visão.

Contudo, outras perspectivas podem ser alvo de modelos. A abordagem de Engenharia de Requisitos Baseada em Objetivos (*Goal-Oriented Requirements Engineering* - GORE), p.ex., assume que objetivos são uma perspectiva fundamental, pois estabelecem o "porquê" do sistema (e, portanto, dos elementos identificados em outras perspectivas). Na abordagem GORE, as razões para um novo sistema (ou uma nova

versão de um sistema) precisam ser explicitadas em termos de objetivos a serem satisfeitos por ele (LAMSWEERDE, 2009) e, para tal, modelos de objetivos devem ser desenvolvidos.

A análise de requisitos é uma atividade extremamente vinculada ao levantamento de requisitos. Durante o levantamento de requisitos, alguns problemas são identificados e tratados. Entretanto, determinados problemas somente são identificados por meio de uma análise mais detalhada. A análise de requisitos ajuda a entender e detalhar os requisitos levantados, a descobrir problemas nesses requisitos e a obter a concordância sobre as alterações, de modo a satisfazer a todos os envolvidos. Seu objetivo é estabelecer um conjunto acordado de requisitos completos, consistentes e sem ambiguidades, que possa ser usado como base para as demais atividades do processo de desenvolvimento de software (KOTONYA; SOMMERVILLE, 1998).

De forma resumida, pode-se dizer que a análise atende a dois propósitos principais: (i) prover uma base para o entendimento e concordância entre clientes e desenvolvedores sobre o que o sistema deve fazer e (ii) prover uma especificação que guie os desenvolvedores na demais etapas do desenvolvimento, sobretudo no projeto, implementação e testes do sistema (PFLEEGER, 2004).

O processo de ER é dominado por fatores humanos, sociais e organizacionais. Ele envolve pessoas de diferentes áreas de conhecimento e com objetivos individuais e organizacionais diferentes. Dessa forma, é comum que cada indivíduo tente influenciar os requisitos para que seu objetivo seja alcançado, sem necessariamente alcançar os objetivos dos demais (KOTONYA; SOMMERVILLE, 1998).

Problemas e conflitos encontrados nos requisitos devem ser listados. Usuários, clientes, especialistas de domínio e engenheiros de requisitos devem discutir os requisitos que apresentam problemas, negociar e chegar a um acordo sobre as modificações a serem feitas. Idealmente, as discussões devem ser governadas pelas necessidades da organização, incluindo o orçamento e o cronograma disponíveis. No entanto, muitas vezes, as negociações são influenciadas por considerações políticas e os requisitos são definidos em função da posição e da personalidade dos indivíduos e não em função de argumentos e razões (KOTONYA; SOMMERVILLE, 1998).

A maior parte do tempo da negociação é utilizada para resolver conflitos de requisitos. Quando discussões informais entre analistas, especialistas de domínio e usuários não forem suficientes para resolver os problemas, é necessária a realização de reuniões de negociação, que envolvem (KOTONYA; SOMMERVILLE, 1998):

- Discussão: os requisitos que apresentam problemas são discutidos e os interessados presentes opinam sobre eles.
- Priorização: requisitos são priorizados para identificar requisitos críticos e ajudar nas decisões e planejamento.
- Concordância: soluções para os problemas são identificadas, mudanças são feitas e um acordo sobre o conjunto de requisitos é acertado.

2.2.3 – Documentação de Requisitos

Os requisitos e modelos capturados nas etapas anteriores devem ser descritos e apresentados em documentos. A documentação é, portanto, uma atividade de registro e oficialização dos resultados da Engenharia de Requisitos. Como resultado, um ou mais documentos devem ser produzidos.

Uma boa documentação fornece muitos benefícios, tais como (IEEE, 1998; NUSEIBEH; EASTERBROOK, 2000): (i) facilita a comunicação dos requisitos; (ii) reduz o esforço de desenvolvimento, pois sua preparação força usuários e clientes a considerar os requisitos atentamente, evitando retrabalho nas fases posteriores; (iii) fornece uma base realística para estimativas; (iv) fornece uma base para verificação e validação; (v) serve como base para futuras manutenções ou incremento de novas funcionalidades.

A documentação dos requisitos tem um conjunto diversificado de interessados, dentre eles (SOMMERVILLE, 2007; KOTONYA; SOMMERVILLE, 1998):

- Clientes, Usuários e Especialistas de Domínio: são interessados na documentação de requisitos, uma vez que atuam na especificação, avaliação e alteração de requisitos.
- Gerentes de Cliente: utilizam o documento de requisitos para planejar um pedido de proposta para o desenvolvimento de um sistema, contratar um fornecedor e para acompanhar o desenvolvimento do sistema.
- Gerentes de Fornecedor: utilizam a documentação dos requisitos para planejar uma proposta para o sistema e para planejar e acompanhar o processo de desenvolvimento.
- Desenvolvedores (analistas, projetistas, programadores e mantenedores): utilizam a documentação dos requisitos para compreender o sistema e as relações entre suas partes.
- Testadores: utilizam a documentação dos requisitos para projetar casos de teste, sobretudo testes de validação do sistema.

Diferentes interessados têm propósitos diferentes. Assim, pode ser útil ter mais do que um documento para registrar os resultados da engenharia de requisitos. Conforme discutido anteriormente, Pfleeger (2004) sugere que dois tipos de documentos de requisitos sejam elaborados: um Documento de Definição de Requisitos e um Documento de Especificação de Requisitos.

O Documento de Definição de Requisitos deve conter uma descrição do propósito do sistema, uma breve descrição do domínio do problema tratado pelo sistema e listas de requisitos funcionais e não funcionais, descritos em linguagem natural (requisitos de cliente). Para facilitar a identificação e rastreamento dos requisitos, devem-se utilizar identificadores únicos para cada um dos requisitos listados. O público-alvo desse documento são clientes, usuários, gerentes (de cliente e de fornecedor) e desenvolvedores.

O Documento de Especificação de Requisitos deve conter os requisitos escritos a partir da perspectiva do desenvolvedor, devendo haver uma correspondência direta com os requisitos no Documento de Definição de Requisitos, de modo a se ter requisitos rastreáveis. Os vários modelos produzidos na fase de análise devem ser apresentados no

Documento de Especificação de Requisitos, bem como glossários de termos usados e outras informações julgadas relevantes.

Deve-se observar que não há um padrão definido quanto à quantidade e ao nome dos documentos de requisitos. Há organizações que optam por ter apenas um documento de requisitos, contendo diversas seções, algumas tratando de requisitos de cliente, outras tratando de requisitos de sistema. Outras organizações, por sua vez, fazem uso de vários documentos distintos, capturando em documentos separados, por exemplo, requisitos funcionais, requisitos não funcionais, modelos de caso de uso e modelos estruturais e comportamentais. De fato, cabe a cada organização definir a quantidade, o nome e o conteúdo de cada documento. Para estruturar o conteúdo, é necessário que a organização defina seus modelos de documentos de requisitos.

Várias diretrizes têm sido propostas com objetivo de melhorar a estrutura e organização da documentação de requisitos, dentre elas (SOMMERVILLE; SAWYER, 1997; KOTONYA; SOMMERVILLE, 1998; PRESSMAN, 2006; WIEGERS, 2003): (i) definir um modelo de documento (*template*) para cada tipo de documento a ser considerado, definindo um padrão de estrutura para o documento; (ii) explicar como cada classe de leitores deve usar os diferentes tipos de documentos; (iii) definir termos especializados em um glossário; (iv) organizar o layout do documento para facilitar a leitura; (v) auxiliar os leitores a encontrar a informação, incluindo recursos tais como listas de conteúdo e índices, e organizando os requisitos em capítulos, seções e subseções identificadas; (vi) identificar as fontes dos requisitos de modo a manter dados da origem do requisito, de modo que, quando alguma mudança for solicitada, seja possível saber com quem essa mudança deve ser discutida e avaliada; e (vii) criar um identificador único para cada requisito, de modo a facilitar a rastreabilidade e o controle de mudanças.

2.2.4 – Verificação e Validação de Requisitos

As atividades de Verificação & Validação (V&V) devem ser iniciadas o quanto antes no processo de desenvolvimento de software, pois quanto mais tarde os defeitos são encontrados, maiores os custos associados à sua correção (ROCHA; MALDONADO; WEBER, 2001). Uma vez que os requisitos são a base para o desenvolvimento, é fundamental que eles sejam cuidadosamente avaliados. Assim, os documentos produzidos durante a atividade de documentação de requisitos devem ser submetidos à verificação e à validação.

É importante realçar a diferença entre verificação e validação. O objetivo da verificação é assegurar que o software esteja sendo construído de forma correta. Deve-se verificar se os artefatos produzidos atendem aos requisitos estabelecidos e se os padrões organizacionais (de produto e processo) foram consistentemente aplicados. Por outro lado, o objetivo da validação é assegurar que o software que está sendo desenvolvido é o software correto, ou seja, assegurar que os requisitos, e o software deles derivado, atendem ao uso proposto (ROCHA; MALDONADO; WEBER, 2001).

No caso de requisitos, a verificação é feita, sobretudo, em relação à consistência entre requisitos e modelos e à conformidade com padrões organizacionais de documentação de requisitos. Já a validação tem de envolver a participação de usuários e clientes, pois somente eles são capazes de dizer se os requisitos atendem aos propósitos do sistema.

Nas atividades de V&V de requisitos, examinam-se os documentos de requisitos para assegurar que (PRESSMAN, 2006; KOTONYA; SOMMERVILLE, 1998; WIEGERS, 2003): (i) todos os requisitos do sistema tenham sido declarados de modo não-ambíguo, (ii) as inconsistências, conflitos, omissões e erros tenham sido detectados e corrigidos, (iii) os documentos estão em conformidade com os padrões estabelecidos e (iv) os requisitos realmente satisfazem às necessidades dos clientes e usuários. Em outras palavras, idealmente, um requisito, seja ele funcional ou não funcional, deve ser (WIEGERS, 2003; PFLEEGER, 2004):

- Completo: o requisito deve descrever completamente a funcionalidade a ser entregue (no caso de requisito funcional) ou a restrição a ser considerada (no caso de requisito não funcional). Ele deve conter as informações necessárias para que o desenvolvedor possa projetar, implementar e testar essa funcionalidade ou restrição.
- Correto: cada requisito deve descrever exatamente a funcionalidade ou restrição a ser incorporada ao sistema.
- Consistente: o requisito não deve ser ambíguo ou conflitar com outro requisito.
- Realista: deve ser possível implementar o requisito com a capacidade e com as limitações do sistema e do ambiente de desenvolvimento.
- Necessário: o requisito deve descrever algo que o cliente realmente precisa ou que é requerido por algum fator externo ou padrão da organização.
- Passível de ser priorizado: os requisitos devem ter ordem de prioridade para facilitar o gerenciamento durante o desenvolvimento do sistema.
- Verificável e passível de confirmação: deve ser possível desenvolver testes para verificar se o requisito foi realmente implementado.
- Rastreável: deve ser possível identificar quais requisitos foram tratados em um determinado artefato, bem como identificar que produtos foram originados a partir de um requisito.

Neste contexto é útil (WIEGERS, 2003):

- Realizar revisões dos documentos de requisitos, procurando por problemas (conflitos, omissões, inconsistências, desvios dos padrões etc) e discutindo soluções.
- Definir casos de teste para os requisitos especificados.
- Definir critérios de aceitação de requisitos, i.e., os usuários devem descrever como vão determinar se o produto atende às suas necessidades e se é adequado para uso.

De maneira geral, i.e., sem estarem restritas a requisitos, as atividades de V&V envolvem análises estáticas e dinâmicas. A análise dinâmica (ou testes) objetiva detectar defeitos ou erros no software por meio da execução do produto. Já a análise estática não envolve a execução do produto, sendo feita por meio de revisões dos artefatos a serem avaliados (ROCHA; MALDONADO; WEBER, 2001). No caso de requisitos, podem-se realizar revisões dos documentos de requisitos para avaliar requisitos e modelos (análise

estática), bem como é possível utilizar prototipagem para validar requisitos (análise dinâmica).

A análise dinâmica, neste caso, se justifica, pois, muitas vezes, as pessoas encontram dificuldades em visualizar como os requisitos serão traduzidos em um sistema. Essa dificuldade pode ser amenizada por meio de protótipos, que auxiliam os usuários na identificação de problemas e na sugestão de melhorias dos requisitos. Dessa forma, a prototipagem pode ser utilizada no processo de validação de requisitos. Entretanto, sua utilização nessa fase tem uma relação custo-benefício mais efetiva quando ela tiver sido empregada também na fase de levantamento de requisitos (KOTONYA; SOMMERVILLE, 1998).

Das atividades de verificação e validação, a atividade de teste é considerada um elemento crítico para a garantia da qualidade dos artefatos produzidos ao longo do desenvolvimento e, por conseguinte, do produto de software final (ROCHA; MALDONADO; WEBER, 2001). Contudo, exceto por meio de protótipos ou especificações de requisitos executáveis (estas um recurso muito pouco utilizado na prática), não é possível testar requisitos. Entretanto, uma das características de qualidade de um requisito bem elaborado é ser testável e, portanto, uma boa maneira de identificar problemas nos requisitos é definir casos de teste para os mesmos. Se um requisito está incompleto, inconsistente ou ambíguo, pode ser difícil definir casos de teste para ele (KOTONYA; SOMMERVILLE, 1998).

Definir casos de teste para requisitos e avaliar protótipos são importantes meios de se verificar e validar requisitos. Contudo, é imprescindível, ainda, realizar revisões dos documentos de requisitos. De maneira geral, em uma revisão, processos, documentos e outros artefatos são revisados por um grupo de pessoas, com o objetivo de avaliar se os mesmos estão em conformidade com os padrões organizacionais estabelecidos e se o propósito de cada um deles está sendo atingido. Assim, o objetivo de uma revisão é detectar erros e inconsistências em artefatos e processos, sejam eles relacionados à forma, sejam eles relacionados ao conteúdo, e apontá-los aos responsáveis pela sua elaboração (ROCHA; MALDONADO; WEBER, 2001).

Em um formato de revisão técnica formal, o processo de revisão começa com o planejamento da revisão, quando uma equipe de revisão é formada, tendo à frente um líder. A equipe de revisão deve incluir membros da equipe que possam ser efetivamente úteis para atingir o objetivo da revisão. Muitas vezes, a pessoa responsável pela elaboração do artefato a ser revisado integra a equipe de revisão (ROCHA; MALDONADO; WEBER, 2001).

O propósito da revisão deve ser previamente informado e o material a ser revisado deve ser entregue com antecedência para que cada membro da equipe de revisão possa avaliá-lo. Uma vez que todos estejam preparados, uma reunião é convocada pelo líder. Dando início à reunião de revisão, normalmente, o autor do artefato apresenta o mesmo e descreve a perspectiva utilizada para a sua construção. O líder orientará o processo de revisão, passando por todos os aspectos relevantes a serem revistos. Todas as considerações dos vários membros da equipe de revisão devem ser discutidas e as decisões registradas, dando origem a uma ata de reunião de revisão, contendo uma lista de defeitos encontrados. Essa reunião deve ser relativamente breve (duas horas, no máximo), uma vez que todos já devem estar preparados para a mesma (ROCHA; MALDONADO; WEBER, 2001).

No que se refere à revisão de requisitos, diversas técnicas de leitura podem ser usadas. A mais simples é a leitura *ad-hoc*, na qual os revisores aplicam seus próprios conhecimentos na revisão dos documentos de requisitos. Diferentemente de uma abordagem *ad-hoc*, outras técnicas buscam aumentar a eficiência dos revisores, direcionando os esforços para as melhores práticas de detecção de defeitos. Técnicas de leitura baseada em listas de verificação (*checklists*), leitura baseada em perspectivas e leitura de modelos orientados a objetos são bastante usadas na verificação e validação de documentos de requisitos.

Checklists definem uma lista de aspectos que devem ser verificados pelos revisores, guiando-os no trabalho de revisão. Podem ser usados em conjunto com outras técnicas, tais como as técnicas de leitura baseada em perspectiva e leitura de modelos orientados a objetos. A Figura 2.3 mostra um exemplo de *checklist* de requisitos.

Checklist para Avaliação de Documentos de Requisitos			
Projeto:	<<nome do projeto>>		
Documento:	<<nome do documento>>	**Versão:**	<<número>>
Item a ser avaliado		**Resultado**	**Problemas Detectados**
Aderência à Estrutura do Modelo de Documento de Requisitos			
1. O documento segue os padrões de fonte e espaçamento?			
2. Todas as seções obrigatórias estão no documento?			
3. As seções do documento estão consistentes com as orientações do *template* para a elaboração das mesmas?			
Conteúdo do Documento			
1. A descrição do propósito está bem colocada, i.e., é sucinta (um único parágrafo) e descreve o objetivo do sistema?			
2. A descrição do minimundo permite uma compreensão básica do domínio, do negócio e do problema sendo tratado pelo sistema?			
3. Os requisitos listados estão compatíveis com a descrição do minimundo?			
4. Há omissões, inconsistência, informação estranha ou ambiguidade na descrição do minimundo e nos requisitos?			
5. Os requisitos estão sendo descritos em formato apropriado para compreensão por clientes e usuários?			
6. Identificadores de requisitos são únicos?			
7. As descrições dos requisitos seguem os padrões definidos?			
8. As dependências entre requisitos estão adequadamente registradas?			
9. As prioridades dos requisitos estão consistentes?			

Figura 2.3 – Exemplo de Checklist de Requisitos

A técnica de leitura baseada em perspectiva foi desenvolvida especificamente para a verificação e validação de requisitos. Ela explora a observação de quais informações de requisitos são mais ou menos importantes para as diferentes formas de utilização do documento de requisitos. Cada revisor realiza a revisão segundo uma perspectiva diferente. Algumas perspectivas tipicamente consideradas são as perspectivas de clientes, desenvolvedores e testadores. *Checklists* para cada uma dessas perspectivas podem ser providos (ROCHA; MALDONADO; WEBER, 2001).

A leitura de modelos orientados a objetos propõe um conjunto de técnicas de leitura para revisão dos diferentes diagramas utilizados durante um projeto orientado a objetos. Cada uma das técnicas é definida para um conjunto de diagramas a serem

analisados em conjunto. O processo de leitura é realizado de duas maneiras: a leitura horizontal diz respeito à consistência entre artefatos elaborados em uma mesma fase, procurando verificar se esses artefatos estão descrevendo consistentemente diferentes aspectos de um mesmo sistema, no nível de abstração relacionado à fase em questão; a leitura vertical refere-se à consistência entre artefatos elaborados em diferentes fases (análise e projeto) (ROCHA; MALDONADO; WEBER, 2001).

2.2.5 – Gerência de Requisitos

Mudanças nos requisitos ocorrem ao longo de todo o processo de software, desde o levantamento e análise de requisitos até durante a operação do sistema. Elas são decorrentes de diversos fatores, tais como descoberta de erros, omissões, conflitos e inconsistências nos requisitos, melhor entendimento por parte dos usuários de suas necessidades, problemas técnicos, de cronograma ou de custo, mudança nas prioridades do cliente, mudanças no negócio, aparecimento de novos competidores, mudanças econômicas, mudanças na equipe, mudanças no ambiente onde o software será instalado e mudanças organizacionais ou legais. Para minimizar as dificuldades impostas por essas mudanças, é necessário gerenciar requisitos.

O processo de gerência de requisitos envolve as atividades que ajudam a equipe de desenvolvimento a identificar, controlar e rastrear requisitos e gerenciar mudanças de requisitos em qualquer momento ao longo do ciclo de vida do software (KOTONYA; SOMMERVILLE, 1998; PRESSMAN, 2006). Os principais objetivos desse processo são (KOTONYA; SOMMERVILLE, 1998):

- Gerenciar alterações nos requisitos acordados.
- Gerenciar relacionamentos entre requisitos.
- Gerenciar dependências entre requisitos e outros documentos produzidos durante o processo de software.

Para tal, o processo de gerência de requisitos deve incluir as seguintes atividades, ilustradas na Figura 2.4 (WIEGERS, 2003): controle de mudanças, controle de versão, acompanhamento do estado dos requisitos e rastreamento de requisitos.

Figura 2.4 - Atividades da Gerência de Requisitos (WIEGERS, 2003)

O controle de mudança define os procedimentos e padrões que devem ser usados para gerenciar alterações em requisitos, assegurando que qualquer proposta de mudança seja analisada conforme os critérios estabelecidos pela organização (KOTONYA; SOMMERVILLE, 1998). Mudanças podem ser necessárias em diferentes momentos e por diferentes razões. De maneira geral, o controle de mudanças envolve atividades para (KOTONYA; SOMMERVILLE, 1998; WIEGERS, 2003):

- Verificar se uma mudança é válida.
- Descobrir quais os requisitos e artefatos afetados pela mudança, o que envolve rastrear informações.
- Estimar o impacto e o custo das mudanças.
- Negociar as mudanças com os clientes.
- Alterar requisitos e documentos associados.

Para garantir uma abordagem consistente, recomenda-se que as organizações definam um conjunto de políticas de gerência de mudança, contemplando (KOTONYA; SOMMERVILLE, 1998):

- o processo de solicitação de mudança e as informações necessárias para processar cada solicitação;
- o processo de análise de impacto e de custos da mudança, além das informações de rastreabilidade associadas;
- os membros que formalmente avaliarão as mudanças;
- as ferramentas que auxiliarão o processo.

Se as mudanças não forem controladas, alterações com baixa prioridade podem ser implementadas antes de outras mais importantes e modificações com custo alto que não são realmente necessárias podem ser aprovadas.

Ao se detectar a necessidade de alteração em um ou mais requisitos, deve-se registrar uma solicitação de mudança, a qual deve ser avaliada por algum membro da equipe do projeto de software. Nessa avaliação, o impacto da alteração deve ser determinado e valores de custo, esforço, tempo e viabilidade devem ser repassados ao solicitante da mudança. Assim, uma parte crítica do controle de mudanças é a avaliação do impacto de uma mudança no restante do software. Para que seja possível efetuar essa avaliação, cada requisito deve estar identificado unicamente e deve ser possível, por exemplo, saber quais são os requisitos dependentes desse requisito e em quais artefatos do processo de software esse requisito é tratado. Portanto, é necessário estabelecer uma rede de ligações de modo que um requisito e os elementos ligados a ele possam ser rastreados. Surge, então, o conceito de rastreabilidade.

A rastreabilidade pode ser definida como a habilidade de se acompanhar a vida de um requisito em ambas as direções do processo de software e durante todo o ciclo de vida. Ela fornece uma base para o desenvolvimento de uma trilha de auditoria para todo o projeto, possibilitando encontrar outros requisitos e artefatos que podem ser afetados pelas mudanças solicitadas (PALMER, 1997). Para tal, é necessário haver ligações entre requisitos e entre requisitos e outros elementos do processo de software. Assim, a identificação de dependências entre requisitos, de requisitos conflitantes, da origem dos requisitos e de seus interessados, além da identificação de em quais artefatos produzidos durante o desenvolvimento de software um requisito é tratado, é de fundamental

importância para que a rastreabilidade possa ser implementada (WIEGERS, 2003; ROBERTSON; ROBERTSON, 2006; KOTONYA; SOMMERVILLE, 1998).

Para apoiar a gerência de requisitos, matrizes de rastreabilidade podem ser produzidas. Elas relacionam os requisitos identificados a um ou mais aspectos do sistema ou do seu ambiente, de modo que elas possam ser procuradas rapidamente para entender como uma modificação em um requisito vai afetar diferentes aspectos do sistema. Dentre as muitas possíveis matrizes de rastreabilidade, há as seguintes:

- Matriz de rastreabilidade de dependência entre requisitos: indica como os requisitos estão relacionados uns com os outros.
- Matriz de rastreabilidade requisitos ←→ fontes: indica a fonte de cada requisito.
- Matriz de rastreabilidade requisitos ←→ subsistemas: indica os subsistemas que tratam os requisitos.
- Matriz de rastreabilidade requisitos ←→ casos de uso: indica os casos de uso que detalham um requisito funcional ou tratam um requisito não funcional.

Além das matrizes de rastreabilidade de requisitos, outras matrizes de rastreabilidade entre outros artefatos do processo podem ser construídas, de modo a apoiar a gerência de requisitos. Por exemplo, ao se estabelecer uma matriz de rastreabilidade casos de uso ←→ classes, indicando que classes de um modelo de análise são necessárias para se tratar um caso de uso, é possível, em conjunto com uma matriz de rastreabilidade requisitos ←→ casos de uso, saber que classes são importantes no tratamento de um requisito.

Davis (apud KOTONYA; SOMMERVILLE, 1998) aponta quatro tipos de rastreabilidade interessantes para que uma análise de impacto de mudanças possa ser feita mais facilmente, como ilustra a Figura 2.5:

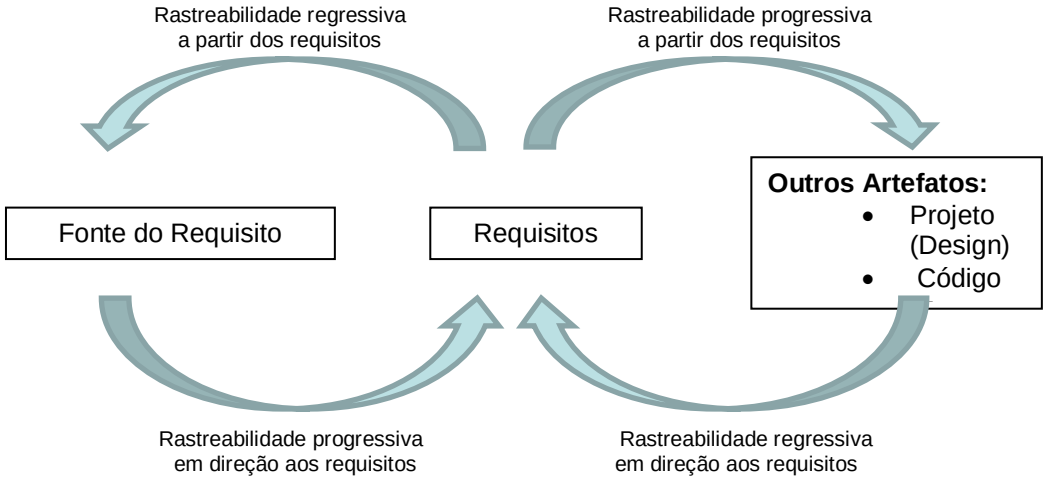

Figura 2.5 – Tipos de Rastreabilidade

- Regressiva a partir dos requisitos (*backward-from traceability*): relaciona requisitos com suas origens/fontes (outros documentos ou pessoas);
- Progressiva a partir dos requisitos (*forward-from traceability*): relaciona requisitos a artefatos do projeto (modelos de análise e projeto, código etc.);

- Regressiva em direção aos requisitos (*backward-to traceability*): relaciona artefatos do projeto aos requisitos;
- Progressiva em direção aos requisitos (*forward-to traceability*): relaciona as fontes aos requisitos relevantes.

Embora a rastreabilidade de requisitos não possa ser completamente automatizada, porque o conhecimento das ligações se origina na mente dos membros da equipe de desenvolvimento, uma vez identificadas essas ligações, ferramentas de apoio são importantíssimas para ajudar a gerenciar a grande quantidade de informações de rastreabilidade (WIEGERS, 2003).

Requisitos guiam várias atividades do processo de software, dentre elas planejamento, projeto (*design*), codificação e testes, e, portanto, esses artefatos devem ser rastreáveis para e a partir dos requisitos. A seguir são citadas algumas situações, em diversas etapas, nas quais os requisitos são importantes (WIEGERS, 2003):

- Planejamento de Projeto: requisitos são úteis para dimensionar o projeto, sendo utilizados para estimar o tamanho do produto. Planos devem ser atualizados caso haja mudanças em requisitos e requisitos prioritários são usados para direcionar iterações, quando modelos de ciclo de vida iterativos são adotados.
- Projeto e Codificação: requisitos, com destaque para os não funcionais, são usados para direcionar o projeto da arquitetura do sistema e são alocados a componentes. Mudanças em requisitos devem ser rastreadas para o projeto e o código gerado, de modo a guiar as alterações.
- Testes: requisitos são a base para diversos tipos de testes, dentre eles testes de sistema e de aceitação.

2.3 – Engenharia de Requisitos e Normas e Modelos de Qualidade

Visando à qualidade no processo de software, modelos e normas de qualidade de processo de software têm sido propostos, dentre eles o CMMI (*Capability Maturity Model Integration*) (SEI, 2010) e o Modelo de Referência MPS para Software (MR-MPS-SW) (SOFTEX, 2016).

O CMMI é um modelo de qualidade de processo desenvolvido pelo Instituto de Engenharia de Software (*Software Engineering Institute* – SEI) da Universidade de Carnegie Mellon. O CMMI para Desenvolvimento (CMMI-Dev) (SEI, 2010) enfoca o processo de software e tem como objetivo fornecer diretrizes para a definição e melhoria de processos de software de uma organização. O CMMI, em sua representação em estágio, possui cinco níveis de maturidade: 1- Inicial, 2- Gerenciado, 3- Definido, 4- Gerenciado Quantitativamente e 5- Em Otimização.

O CMMI-Dev trata de requisitos tanto no nível 2, através da área de processo Gestão de Requisitos, quanto no nível 3, por meio da área de processo Desenvolvimento de Requisitos. Na Gestão de Requisitos o objetivo é gerenciar os requisitos dos produtos e componentes de produto do projeto e garantir alinhamento entre esses requisitos e os planos e produtos de trabalho do projeto (SEI, 2010). Para isso, o CMMI sugere as seguintes práticas:

SG 1 Gerenciar Requisitos
SP 1.1 Entender os requisitos.
SP 1.2 Obter comprometimento com os requisitos.
SP 1.3 Gerenciar mudanças de requisitos.
SP 1.4 Manter rastreabilidade bidirecional dos requisitos.
SP 1.5 Garantir alinhamento entre o trabalho de projeto (planos de projeto e produtos de trabalho) e requisitos.

O Desenvolvimento de Requisitos, por sua vez, visa levantar, analisar e estabelecer requisitos de cliente, do produto e de componentes do produto (SEI, 2010). Neste caso, o CMMI sugere as seguintes práticas:

SG 1 Desenvolver Requisitos de Cliente
SP 1.1 Levantar necessidades.
SP 1.2 Transformar necessidades dos interessados em requisitos de cliente.

SG 2 Desenvolver Requisitos do Produto
SP 2.1 Estabelecer os requisitos do produto e de componentes do produto.
SP 2.2 Alocar requisitos de componentes do produto.
SP 2.3 Identificar requisitos de interface.

SG 3 Analisar e Validar Requisitos
SP 3.1 Estabelecer conceitos e cenários operacionais.
SP 3.2 Estabelecer uma definição da funcionalidade e dos atributos de qualidade requeridos.
SP 3.3 Analisar requisitos.
SP 3.4 Analisar requisitos para balancear.
SP 3.5 Validar requisitos.

O Programa MPS.BR (Melhoria de Processo do Software Brasileiro) (SOFTEX, 2016) tem como objetivo a melhoria de processo do software brasileiro. Uma das metas do programa MPS.BR é definir e aprimorar um modelo de melhoria e avaliação de processo de software, visando preferencialmente às micro, pequenas e médias empresas, de forma a atender as suas necessidades de negócio e ser reconhecido nacional e internacionalmente como um modelo aplicável à indústria de software. O Modelo de Referência MPS para Software (MR-MPS-SW) é desenvolvido no contexto do MPS.BR e sua base técnica é composta pelas normas ISO/IEC 12207 e ISO/IEC 15504-2, além buscar garantir conformidade com o CMMI. Essa abordagem visa garantir que o MR-MPS-SW está em conformidade com padrões internacionais.

Assim como o CMMI, o MR-MPS-SW é organizado em níveis de maturidade. No caso do MR-MPS-SW, contudo, são sete níveis, a saber: G- Parcialmente Gerenciado, F- Gerenciado, E- Parcialmente Definido, D- Largamente Definido, C- Definido, B- Gerenciado Quantitativamente, A- Em Otimização. O MR-MPS-SW procura manter uma correspondência entre seus níveis de maturidade e os níveis de maturidade do CMMI. Assim, o nível 2 do CMMI corresponde ao nível F do MPS, o nível 3 do CMMI ao nível C do MPS, o nível 4 do CMMI ao nível B do MPS e finalmente o nível 5 do CMMI corresponde ao nível A do MPS.

O MR-MPS-SW define, em seu nível G, o processo de Gerência de Requisitos e no nível D o processo de Desenvolvimento de Requisitos. O propósito do processo de

Gerência de Requisitos é "*gerenciar os requisitos do produto e dos componentes do produto do projeto e identificar inconsistências entre os requisitos, os planos do projeto e os produtos de trabalho do projeto*". Esse processo tem como resultados esperados:

> GRE 1. O entendimento dos requisitos é obtido junto aos fornecedores de requisitos;
>
> GRE 2. Os requisitos são avaliados com base em critérios objetivos e um comprometimento da equipe técnica com esses requisitos é obtido;
>
> GRE 3. A rastreabilidade bidirecional entre os requisitos e os produtos de trabalho é estabelecida e mantida;
>
> GRE 4. Revisões em planos e produtos de trabalho do projeto são realizadas visando identificar e corrigir inconsistências em relação aos requisitos;
>
> GRE 5. Mudanças nos requisitos são gerenciadas ao longo do projeto.

O Desenvolvimento de Requisitos, por sua vez, tem por objetivo "*definir os requisitos do cliente, do produto e dos componentes do produto*". Esse processo tem como resultados esperados:

> DRE 1. As necessidades, expectativas e restrições do cliente, tanto do produto quanto de suas interfaces, são identificadas;
>
> DRE 2. Um conjunto definido de requisitos do cliente é especificado e priorizado a partir das necessidades, expectativas e restrições identificadas;
>
> DRE 3. Um conjunto de requisitos funcionais e não-funcionais, do produto e dos componentes do produto que descrevem a solução do problema a ser resolvido, é definido e mantido a partir dos requisitos do cliente;
>
> DRE 4. Os requisitos funcionais e não-funcionais de cada componente do produto são refinados, elaborados e alocados. Interfaces internas e externas do produto e de cada componente do produto são definidas;
>
> DRE 5. Conceitos operacionais e cenários são desenvolvidos;
>
> DRE 6. Os requisitos são analisados, usando critérios definidos, para balancear as necessidades dos interessados com as restrições existentes;
>
> DRE 7. Os requisitos são validados.

De uma maneira geral, pode-se constatar que a Engenharia de Requisitos, pela sua importância no contexto do desenvolvimento de software, é cuidadosamente tratada pelos principais modelos de qualidade de processo de software.

Referências do Capítulo

AURUM, A., WOHLIN, C., *Engineering and Managing Software Requirements*, Springer-Verlag, 2005.

BLAHA, M., RUMBAUGH, J., *Modelagem e Projetos Baseados em Objetos com UML 2*, Elsevier, 2006.

IEEE, IEEE Recommended Practice for Software Requirements Specifications: IEEE Std 830-1998. New York: IEEE, 1998.

KENDALL, K.E., KENDALL, J.E.; *Systems Analysis and Design*, Prentice Hall, 8th Edition, 2010.

van LAMSWEERDE, A., *Requirements Engineering: From System Goals to UML Models to Software Specifications*, Willey, 2009.

KOTONYA, G., SOMMERVILLE, I., *Requirements engineering: processes and techniques*. Chichester, England: John Wiley, 1998.

NUSEIBEH, B., EASTERBROOK, S., "Requirements engineering: a roadmap". In: Proceedings of the Conference on the Future of Software Engineering, Limerick, Ireland, 2000.

PALMER, J.D., "Traceability". In: THAYER, H. R.; DORFMAN, M. (Org.) *Software requeriments engineering*. 2nd edition, Los Alamitos, California: IEEE Computer Society, p.364-374, 1997.

PFLEEGER, S.L., *Engenharia de Software: Teoria e Prática*, São Paulo: Prentice Hall, 2ª edição, 2004.

PRESSMAN, R.S., *Engenharia de Software*, McGraw-Hill, 6ª edição, 2006.

ROBERTSON, S., ROBERTSON, J. *Mastering the Requirements Process***.** 2nd Edition. Addison Wesley, 2006.

ROCHA, A.R.C., MALDONADO, J.C., WEBER, K.C., *Qualidade de Software: Teoria e Prática*. São Paulo: Prentice Hall, 2001.

SEI, CMMI for Development, Version, 1.3, CMMI-Dev V1.3, CMU/SEI-2010-TR-033, 2010.

SOFTEX, MPS.BR – Melhoria de Processo do Software Brasileiro – Guia Geral MPS de Software: 2016, Janeiro 2016.

SOMMERVILLE, I., *Engenharia de Software*, 8ª Edição. São Paulo: Pearson – Addison Wesley, 2007.

SOMMERVILLE, I., SAWYER, P., *Requirements engineering: a good practice guide*. Chichester, England: John Wiley, 1997.

WIEGERS, K.E., *Software Requirements: Practical techniques for gathering and managing requirements throughout the product development cycle*. 2nd Edition, Microsoft Press, Redmond, Washington, 2003.