3.3 – Requisitos e Modelagem de Processos de Negócio

O uso de sistemas de informação tem por objetivo principal apoiar ações dos processos de negócio das organizações. Deve-se ter em mente que o principal interesse dos clientes não é o sistema de informação em si, mas sim os efeitos positivos gerados pela sua utilização. Dentre os principais benefícios da implantação de sistemas de informação para apoiar processos de negócio estão (DAVENPORT, 2000): (i) a automatização de tarefas antes realizadas manualmente, (ii) a racionalização dos dados, (iii) a implementação de melhorias nos processos da organização, (iv) ajuste das interfaces entre áreas, (v) aperfeiçoamento dos serviços aos clientes e (vi) geração de informações gerenciais.

Assim, durante o levantamento de requisitos, é necessário compreender o contexto organizacional, no qual o sistema será inserido, bem como se alinhar aos objetivos desse contexto. Se o entendimento do contexto organizacional não ocorre, os requisitos levantados tendem a ficar mais centrados em aspectos tecnológicos, sem levar em consideração fatores organizacionais, tais como (CARVALHO, 2009):

- Regras de negócio que têm impacto sobre o sistema;
- Usuários que utilizam as informações geradas pelo sistema;
- Impactos gerados pelo sistema na forma de execução dos processos de negócio da organização.

É importante notar que os sistemas de informação habilitam os processos de negócio e que, se os processos de negócio e os sistemas não estiverem alinhados, então os sistemas não atenderão às expectativas de seus usuários. Essa falta de alinhamento é apontada como sendo uma das principais causas do fracasso de projetos de desenvolvimento de sistemas de informação (FRYE; GULLEDDGE, 2007). Assim, levantar requisitos tomando por base processos de negócio é importante, pois sistemas de informação mantêm estreita relação com o ambiente organizacional. Um tratamento dissociado do aparato de levantamento de requisitos da visão de negócios pode levar à subutilização de potencialidades tecnológicas (CARVALHO, 2009).

Um modelo de negócio (*business model*) é, na verdade, um conjunto de modelos que provê uma visualização dos processos de negócio, de como estes são executados, quais são as suas metas, como cada processo trabalha para atingir essas metas, quais as unidades organizacionais e os papéis das pessoas envolvidos em cada atividade do processo, quais as localidades onde a organização está distribuída e quais eventos deflagram seus processos e atividades. Como ilustra a Figura 3.1, para modelar esses diferentes aspectos, diferentes tipos de modelos podem ser elaborados, dentre eles (KNIGHT, 2004):

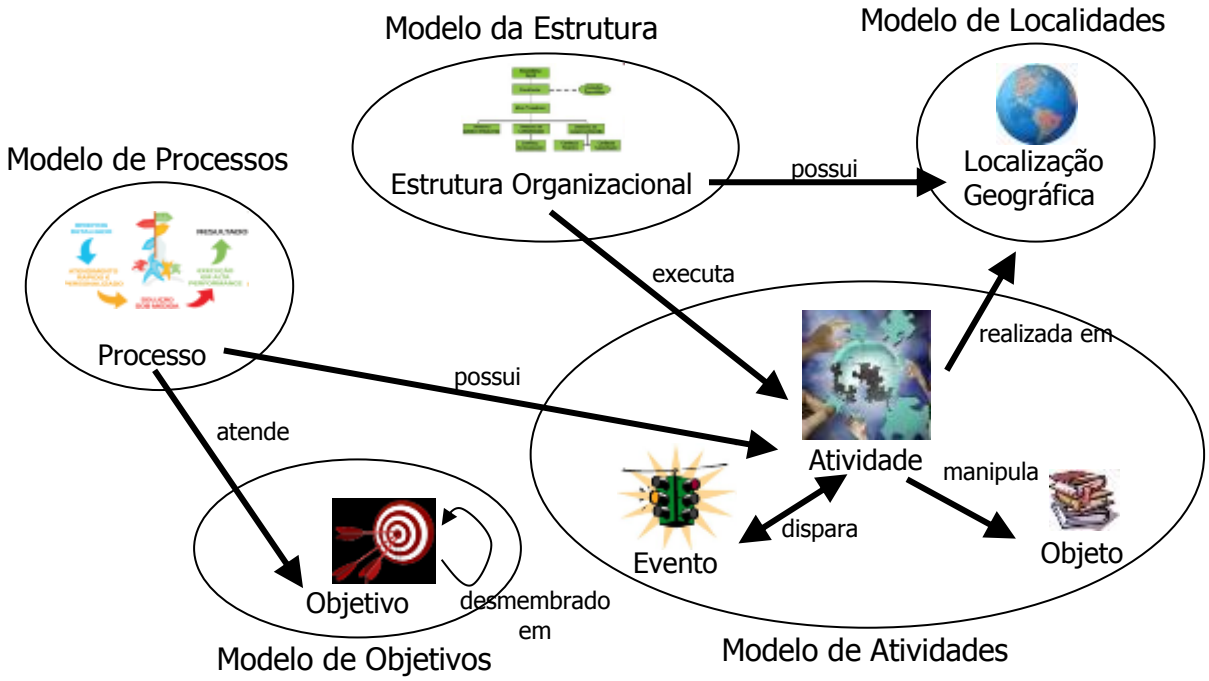

Figura 3.2 – Submodelos do Modelo de Negócio (adaptado de (KNIGHT, 2004)).

- Modelos da Estrutura Organizacional: representam as unidades organizacionais, seus papéis e seus relacionamentos;
- Modelos de Localizações Geográficas: representam as localidades onde a organização está distribuída e os relacionamentos entre localidades e unidades organizacionais;
- Modelos de objetivos: mostram os objetivos da organização e seus relacionamentos, o desdobramento dos objetivos em subobjetivos e o relacionamento entre os objetivos e os processos de negócio;
- Modelos de processos: representam os processos de negócio executados na organização (com suas atividades e relacionamentos) e seus desdobramentos em subprocessos e atividades;
- Modelos de atividades: mostram os relacionamentos entre as atividades dos processos de negócio, seus responsáveis, objetos de negócio e eventos que disparam ou são disparados com a execução das atividades.

Há muitas notações utilizadas para representar os vários tipos de modelos citados anteriormente, sendo que alguns autores propõem o uso de diagramas da UML ou extensões deles para essa finalidade. Em especial, os diagramas de atividades são bastante utilizados para a representação de modelos de processos de negócio e de atividades.

Um dos usos mais comuns de modelos de processo/atividades na Engenharia de Requisitos é a derivação de casos de uso a partir da análise das informações capturadas nos diagramas que representam os aspectos do negócio, incluindo a identificação das atividades dos processos que serão apoiadas pelo sistema, bem como a identificação dos atores dos casos de uso a partir dos executores (pessoas ou máquinas) relacionados aos processos.

No decorrer da modelagem de processos, a reunião dos envolvidos no processo permite que seja estabelecida uma visão abrangente de todo o contexto, possibilitando a identificação de divergências e abrindo espaços para a melhoria, o que pode levar à reengenharia dos

processos de negócio. Processos correntes (processos *AS IS*) podem dar origem a novos processos, mais eficazes, a serem implantados na organização (processos *TO BE*). O contexto no qual o sistema será inserido é considerado como parte integral na aplicação da abordagem orientada a modelos de processos, permitindo que o sistema seja considerado um agente de mudanças e não apenas a automatização das práticas correntes, sejam elas boas ou não.

A modelagem de processos complementa as práticas convencionais de Engenharia de Requisitos, auxiliando o cliente a adquirir maturidade acerca da complexidade do seu próprio negócio e revelando o grau de adequação dos requisitos levantados com os processos (e objetivos) da organização (CARDOSO; ALMEIDA; GUIZZARDI, 2008).

É importante realçar que a modelagem de processos de negócios independe do desenvolvimento de sistemas e pode ser conduzida independentemente para a construção de uma arquitetura organizacional de referência (*enterprise architecture*). Quando necessária a construção de um sistema que dê apoio a partes dos processos modelados nessa arquitetura, é necessário retornar ao cliente somente para levantar requisitos de sistema e não para levantar requisitos de negócio (CARDOSO; ALMEIDA; GUIZZARDI, 2008).

3.4 – Escrevendo e Documentando Requisitos de Cliente

Os resultados do levantamento de requisitos têm de ser registrados em um documento, de modo que possam ser verificados, validados e utilizados como base para outras atividades do processo de software. Para que sejam úteis, os requisitos têm de ser escritos em um formato compreensível por todos os interessados. Além disso, esses interessados devem interpretá-los uniformemente.

Normalmente, requisitos são documentados usando alguma combinação de linguagem natural, modelos, tabelas e outros elementos. A linguagem natural é quase sempre imprescindível, uma vez que é a forma básica de comunicação compreensível por todos os interessados. Contudo, ela geralmente abre espaços para ambiguidades e má interpretação. Assim, é interessante procurar estruturar o uso da linguagem natural e complementar a descrição dos requisitos com outros elementos.

Conforme discutido no Capítulo 2, diferentes abordagens podem ser usadas para documentar requisitos. Neste texto, sugerimos elaborar dois documentos: o Documento de Definição de Requisitos e o Documento de Especificação de Requisitos. O Documento de Definição de Requisitos é mais sucinto, escrito em um nível mais apropriado para o cliente e contempla apenas os requisitos de cliente. O Documento de Especificação de Requisitos é mais detalhado, escrito a partir da perspectiva dos desenvolvedores (PFLEEGER, 2004), normalmente contendo diversos modelos para descrever requisitos de sistema.

Os requisitos de cliente devem ser descritos de modo a serem compreensíveis pelos interessados no sistema que não possuem conhecimento técnico detalhado. Eles devem versar somente sobre o comportamento externo do sistema, em uma linguagem simples, direta e sem usar terminologia específica de software (SOMMERVILLE, 2007).

O Documento de Definição de Requisitos tem como propósito descrever os requisitos de cliente, tendo como público-alvo clientes, usuários, gerentes (de cliente e de fornecedor) e desenvolvedores. Há muitos formatos distintos propostos na literatura para documentos de requisitos. Neste texto, é proposta uma estrutura bastante simples para esse tipo de documento, contendo apenas quatro seções:

- Introdução: breve introdução ao documento, descrevendo seu propósito e estrutura.
- Descrição do Propósito do Sistema: descreve o propósito geral do sistema.
- Descrição do Minimundo: apresenta, em um texto corrido, uma visão geral do domínio, do problema a ser resolvido e dos processos de negócio apoiados, bem como as principais ideias do cliente sobre o sistema a ser desenvolvido.
- Requisitos de cliente: apresenta os requisitos de cliente em linguagem natural.

As três primeiras seções não têm nenhuma estrutura especial, sendo apresentadas na forma de um texto corrido. A introdução deve ser breve e basicamente descrever o propósito e a estrutura do documento, podendo seguir um padrão preestabelecido pela organização. A descrição do propósito do sistema deve ser direta e objetiva, tipicamente em um único parágrafo. Já a descrição do minimundo é um pouco maior, algo entre uma e duas páginas, descrevendo aspectos gerais e relevantes para um primeiro entendimento do domínio, do problema a ser resolvido e dos processos de negócio apoiados. Contém as principais ideias do cliente sobre o sistema a ser desenvolvido, obtidas no levantamento preliminar e exploratório do sistema. Não se devem incluir detalhes.

A Seção 4, por sua vez, não deve ter um formato livre. Ao contrário, deve seguir um formato estabelecido pela organização, contendo, dentre outros: identificador do requisito, descrição, tipo, origem, prioridade, responsável, interessados, dependências em relação a outros requisitos e requisitos conflitantes. A definição de padrões organizacionais para a definição de requisitos é essencial para garantir uniformidade e evitar omissão de informações importantes acerca dos requisitos (SOMMERVILLE, 2007; WIEGERS, 2003). Como consequência, o padrão pode ser usado como um guia para a verificação de requisitos. A Tabela 3.4 apresenta um exemplo de padrão tabular. Sugerem-se agrupar requisitos de um mesmo tipo em uma mesma tabela. Assim, a informação do tipo do requisito não aparece explicitamente no padrão proposto. Além disso, informações complementares podem ser adicionadas em função do tipo de requisito. A seguir, discute-se como cada um dos itens da tabela pode ser tratado, segundo uma perspectiva geral. Na sequência, são tecidas considerações mais específicas sobre a descrição dos diferentes tipos de requisitos.

Tabela 3.4 – Tabela de Requisitos.

Identificador	Descrição	Origem	Prioridade	Responsável	Interessados	Dependências	Conflitos

Os requisitos devem possuir identificadores únicos para permitir a identificação e o rastreamento na gerência de requisitos. Há diversas propostas de esquemas de rotulagem de requisitos. Neste texto, recomenda-se usar um esquema de numeração sequencial por tipo de requisito, sendo usados os seguintes prefixos para designar os diferentes tipos de requisitos: RF – requisitos funcionais; e RNF – requisitos não funcionais. Para outros esquemas de rotulagem, vide (WIEGERS, 2003). É importante destacar que, quando um requisito é eliminado, seu identificador não pode ser atribuído a outro requisito.

A descrição do requisito normalmente é feita na forma de uma sentença em linguagem natural. Ainda que expressa em linguagem natural, é importante adotar um estilo consistente e usar a terminologia do usuário ao invés do jargão típico da computação. Em relação ao estilo, recomenda-se utilizar sentenças em um dos seguintes formatos para descrever requisitos funcionais e não funcionais:

- *O sistema deve <verbo indicando ação, seguido de complemento>*: use o verbo *dever* para designar uma função ou característica requerida para o sistema, ou seja, para descrever um requisito obrigatório. Exemplos: O sistema deve efetuar o controle dos clientes da empresa. O sistema deve processar um pedido do cliente em um tempo inferior a cinco segundos, contado a partir da entrada de dados.
- *O sistema pode <verbo indicando ação, seguido de complemento>*: use o verbo *poder* para designar uma função ou característica desejável para o sistema, ou seja, para descrever um requisito desejável, mas não essencial. Exemplos: O sistema pode notificar usuários em débito. O sistema pode sugerir outros produtos para compra, com base em produtos colocados no carrinho de compras do usuário.

Em algumas situações, pode-se querer explicitar que alguma funcionalidade ou característica não deve ser tratada pelo sistema (dito requisito negativo). Isso pode ser feito indicando-se uma sentença com a seguinte estrutura: *O sistema não deve <verbo indicando ação, seguido de complemento>*. Também é possível registrar que alguma tarefa específica relacionada ao sistema não fará parte do escopo do projeto, tal como a carga de dados de um sistema existente. Nestes casos, pode ser útil ter uma tabela de requisitos negativos.

Wiegers (2003) recomenda diversas diretrizes para a redação de requisitos, dentre elas:

- Escreva frases completas, com a gramática, ortografia e pontuação correta. Procure manter frases e parágrafos curtos e diretos.
- Use os termos consistentemente. Defina-os em um glossário.
- Prefira a voz ativa (o sistema deve fazer alguma coisa) à voz passiva (alguma coisa deve ser feita).
- Sempre que possível, identifique o tipo de usuário. Evite descrições genéricas como "o usuário deve [...]". Se o usuário no caso for, por exemplo, o caixa do banco, indique claramente "o caixa do banco deve [...]".
- Evite termos vagos, que conduzam a requisitos ambíguos e não testáveis, tais como "rápido", "adequado", "fácil de usar" etc.
- Escreva requisitos em um nível consistente de detalhe. Evite requisitos desnecessariamente pequenos, tais como requisitos individuais para tratar de ações de inclusão, remoção, alteração e consulta de um elemento de informação (p.ex., "O sistema deve permitir a alteração de dados de clientes"). Em casos como este, considere agrupar os requisitos menores em um requisito maior (tal como "O sistema deve permitir o controle de clientes"). Por outro lado, evite longos parágrafos narrativos que contenham múltiplos requisitos. Divida um requisito desta natureza em vários.
- O nível de descrição de um requisito deve ser tal que, se o requisito é satisfeito, a necessidade do cliente é atendida. Evite restringir desnecessariamente o projeto (design).
- Escreva requisitos individualmente testáveis. Um requisito bem escrito deve permitir a definição de um pequeno conjunto de testes para verificar se o requisito foi corretamente implementado.

Conforme apontado anteriormente, requisitos devem ser testáveis. Robertson e Robertson (2006) sugerem três maneiras de tornar os requisitos testáveis:

1. Especificar uma descrição quantitativa de cada advérbio ou adjetivo, de modo que o significado dos qualificadores fique claro e não ambíguo.
2. Trocar os pronomes pelos nomes das entidades.
3. Garantir que todo termo importante seja definido em um glossário no documento de requisitos.

A origem de um requisito deve apontar a partir de que entidade (pessoa, documento, atividade) o requisito foi identificado. Um requisito identificado durante uma investigação de documentos, p.ex., tem como origem o(s) documento(s) inspecionado(s). Já um requisito levantado em uma entrevista com certo usuário tem como origem o próprio usuário. A informação de origem é importante para se conseguir rastrear requisitos para a sua origem, prática muito recomendada na Gerência de Requisitos.

Requisitos podem ter importância relativa diferente em relação a outros requisitos. Assim, é importante que o cliente e outros interessados estabeleçam conjuntamente a prioridade de cada requisito.

É muito importante saber quem é o analista responsável por um requisito, bem como quem são os interessados (clientes, usuários etc.) naquele requisito. São eles que estarão envolvidos nas discussões relativas ao requisito, incluindo a tentativa de acabar com conflitos e a definição de prioridades. Assim, deve-se registrar o nome e o papel do responsável e dos interessados em cada requisito.

Um requisito pode depender de outros ou conflitar com outros. Quando as dependências e conflitos forem detectados, devem-se listar os respectivos identificadores nas colunas de dependências e conflitos.

3.4.1 - Escrevendo Requisitos Funcionais

As diretrizes apresentadas anteriormente aplicam-se integralmente a requisitos funcionais. Assim, não há outras diretrizes específicas para os requisitos funcionais. Deve-se realçar apenas que, quando expressos como requisitos de cliente, requisitos funcionais são geralmente descritos de forma abstrata, não cabendo neste momento entrar em detalhes. Detalhes vão ser naturalmente adicionados quando esses mesmos requisitos forem descritos na forma de requisitos de sistema.

Uma alternativa largamente empregada para especificar requisitos funcionais no nível de requisitos de sistema é a modelagem. Uma das técnicas mais comumente utilizadas para descrever requisitos funcionais como requisitos de sistema é a modelagem de casos de uso.

Vale ressaltar que os processos de negócio a serem apoiados pelo sistema tipicamente dão origem a requisitos funcionais. Assim, a partir de uma descrição de minimundo ou em um relatório proveniente de alguma atividade de levantamento de requisitos, podem ser encontrados requisitos funcionais a partir da identificação dos processos a serem apoiados. Além disso, o controle de informações que o negócio precisa gerenciar para apoiar os processos de negócio também deve dar origem a requisitos funcionais representando atividades custodiais (cadastros).

3.4.2 - Escrevendo Requisitos Não Funcionais

Clientes e usuários naturalmente enfocam a especificação de requisitos funcionais. Entretanto para um sistema ser bem-sucedido, é necessário mais do que entregar a funcionalidade correta. Usuários também têm expectativas sobre quão bem o sistema vai funcionar. Características que entram nessas expectativas incluem: quão fácil é usar o sistema, quão rapidamente ele roda, com que frequência ele falha e como ele trata condições inesperadas. Essas características, conhecidas como atributos de qualidade do produto de software, são parte dos requisitos não funcionais do sistema. Essas expectativas de qualidade do produto têm de ser exploradas durante o levantamento de requisitos (WIEGERS, 2003).

Clientes geralmente não apontam suas expectativas de qualidade explicitamente. Contudo, informações providas por eles durante o levantamento de requisitos fornecem algumas pistas sobre o que eles têm em mente. Assim, é necessário definir o que os usuários pensam quando eles dizem que o sistema deve ser amigável, rápido, confiável ou robusto (WIEGERS, 2003).

Há muitos atributos de qualidade que podem ser importantes para um sistema. Uma boa estratégia para levantar requisitos não funcionais de produto consiste em explorar uma lista de potenciais atributos de qualidade que a grande maioria dos sistemas deve apresentar em algum nível. Por exemplo, o modelo de qualidade externa e interna de produtos de software definido na norma ISO/IEC 25010, utilizado como referência para a avaliação de produtos de software, define oito características de qualidade, desdobradas em subcaracterísticas, a saber (ISO/IEC, 2011):

- Aptidão Funcional (*Functional Suitability*): grau em que o produto provê funções que satisfazem às necessidades explícitas e implícitas, quando usado em condições especificadas. Inclui subcaracterísticas que evidenciam a existência de um conjunto de funções e suas propriedades específicas, a saber:
  - Completude Funcional: capacidade do produto de software de prover um conjunto apropriado de funções para tarefas e objetivos do usuário especificados. Refere-se às necessidades declaradas. Um produto no qual falte alguma função requerida não apresenta este atributo de qualidade.
  - Correção Funcional (ou Acurácia): grau em que o produto de software fornece resultados corretos e precisos, conforme acordado. Um produto que apresente dados incorretos, ou com a precisão abaixo dos limites definidos como toleráveis, não apresenta este atributo de qualidade.
  - Adequação Funcional (*Functional Appropriateness*): capacidade do produto de software de facilitar a realização das tarefas e objetivos do usuário. Refere-se às necessidades implícitas.
- Confiabilidade: grau em que o produto executa as funções especificadas com um comportamento consistente com o esperado, por um período de tempo. A confiabilidade está relacionada com os defeitos que um produto apresenta e como este produto se comporta em situações consideradas fora do normal. Suas subcaracterísticas são:
  - Maturidade: é uma medida da frequência com que o produto de software apresenta defeitos ao longo de um período estabelecido de tempo. Refere-se, portanto, à capacidade do produto de software de evitar falhas decorrentes de defeitos no software, mantendo sua operação normal ao longo do tempo.

  - Disponibilidade: capacidade do produto de software de estar operacional e acessível quando seu uso for requerido.
  - Tolerância a falhas: capacidade do produto de software de operar em um nível de desempenho especificado em casos de defeitos no software ou no hardware. Esta subcaracterística tem a ver com a forma como o software reage quando ocorre uma situação externa fora do normal (p.ex., conexão com a Internet interrompida).
  - Recuperabilidade: capacidade do produto de software de se colocar novamente em operação, restabelecendo seu nível de desempenho especificado, e recuperar os dados diretamente afetados no caso de uma falha.

- Usabilidade: grau em que o produto apresenta atributos que permitem que o mesmo seja entendido, aprendido e usado, e que o tornem atrativo para o usuário. Tem como subcaracterísticas:
  - Reconhecimento da Adequação: grau em que os usuários reconhecem que o produto de software é adequado para suas necessidades.
  - Capacidade de Aprendizado: refere-se à facilidade de o usuário entender os conceitos chave do produto e aprender a utilizá-lo, tornando-se competente em seu uso.
  - Operabilidade: refere-se à facilidade do usuário operar e controlar o produto de software.
  - Proteção contra Erros do Usuário: capacidade do produto de software de evitar que o usuário cometa erros.
  - Estética da Interface com o Usuário: capacidade do produto de software de ser atraente ao usuário, lhe oferecendo uma interface com interação agradável.
  - Acessibilidade: capacidade do produto de software ser utilizado por um amplo espectro de pessoas, incluindo portadores de necessidades especiais e com limitações associadas à idade.
- Eficiência de Desempenho: capacidade de o produto manter um nível de desempenho apropriado em relação aos recursos utilizados em condições estabelecidas. Inclui subcaracterísticas que evidenciam o relacionamento entre o nível de desempenho do software e a quantidade de recursos utilizados, a saber:
  - Comportamento em Relação ao Tempo: capacidade do produto de software de fornecer tempos de resposta e de processamento apropriados, quando o software executa suas funções, sob condições estabelecidas.
  - Utilização de Recursos: capacidade do produto de software de usar tipos e quantidades apropriados de recursos, quando executa suas funções, sob condições estabelecidas.
  - Capacidade: refere-se ao grau em que os limites máximos dos parâmetros do produto de software (p.ex., itens que podem ser armazenados, número de usuários concorrentes etc.) atendem às condições especificadas.

- Segurança: grau em que informações e dados são protegidos contra acesso por pessoas ou sistemas não autorizados, bem como grau em que essas informações e dados são disponibilizados para as pessoas ou sistemas com acesso autorizado. Tem como subcaracterísticas:
  - Confidencialidade: grau em que o produto ou sistema garante que os dados estão acessíveis apenas para aqueles autorizados a acessá-los.
  - Integridade: grau em que o sistema, produto ou componente evita acesso não autorizado a, ou modificação de, programas ou dados.
  - Não Repúdio: grau em que se pode provar que ações ou eventos aconteceram, de modo que eles não possam ser repudiados posteriormente.
  - Responsabilização: grau em que as ações de uma entidade (pessoa ou sistema) podem ser rastreadas unicamente a essa entidade.
  - Autenticidade: grau em que se pode provar que a identidade de um sujeito ou recurso é aquela reivindicada.
- Compatibilidade: capacidade do produto de software de trocar informações com outras aplicações e/ou compartilhar o mesmo ambiente de hardware ou software. Suas subcaracterísticas são:
  - Coexistência: grau em que um produto pode desempenhar eficientemente suas funções requeridas, ao mesmo tempo em que compartilha um ambiente e recursos comuns com outros produtos, sem prejuízo para qualquer outro.
  - Interoperabilidade: capacidade do produto de software de interagir com outros sistemas especificados, trocando e usando as informações trocadas.
- Manutenibilidade: capacidade do produto de software de ser modificado. Tem como subcaracterísticas:
  - Modularidade: grau em que um sistema ou programa é composto por componentes discretos (coesos e fracamente acoplados), de modo que mudanças em um componente tenham impacto mínimo sobre os outros.
  - Reusabilidade: capacidade dos componentes do produto de software serem utilizados na construção de outros componentes ou sistemas.
  - Analisabilidade: grau de eficácia e eficiência com que é possível: avaliar o impacto de uma alteração pretendida no produto ou sistema; diagnosticar deficiências ou causas de falhas no produto, ou identificar partes a serem modificadas.
  - Modificabilidade: grau em que um produto ou sistema pode ser eficaz e eficientemente modificado sem introduzir defeitos ou degradar sua qualidade.
  - Testabilidade: grau de eficácia e eficiência com que critérios de teste podem ser estabelecidos para um sistema, produto ou componente, e testes podem ser realizados para determinar se esses critérios foram satisfeitos.

- Portabilidade: refere-se à capacidade do software ser transferido de um ambiente de hardware, software ou operacional para outro. Tem como subcaracterísticas:
    - Adaptabilidade: grau em que um produto ou sistema pode ser eficaz e eficientemente adaptado para ambientes diferentes, ou em evolução, de hardware, software ou operacional.
    - Capacidade para ser instalado (instalabilidade): grau de eficácia e eficiência com que um produto ou sistema pode ser instalado e/ou desinstalado com sucesso em um ambiente especificado.
    - Capacidade de substituição (substituibilidade): grau em que um produto pode substituir outro produto especificado para a mesma finalidade, no mesmo ambiente. Considera, também, a facilidade de atualização para novas versões.

A característica de aptidão funcional e suas subcaracterísticas estão claramente relacionadas a requisitos funcionais. As demais, contudo, podem ser vistas como requisitos não funcionais que quaisquer produtos de software terão de tratar em alguma extensão.

Outro ponto importante a destacar é que diferentes autores listam diferentes características de qualidade, usando classificações próprias. Por exemplo, Bass, Clements e Kazman (2003) consideram, dentre outros, os seguintes atributos de qualidade:

- Disponibilidade: refere-se a falhas do sistema e suas consequências associadas. Uma falha ocorre quando o sistema não entrega mais um serviço consistente com sua especificação.
- Modificabilidade: diz respeito ao custo de modificação do sistema.
- Desempenho: refere-se a tempo.
- Segurança: está relacionada à habilidade do sistema impedir o uso não autorizado, enquanto ainda provê seus serviços para os usuários legítimos.
- Testabilidade: refere-se ao quão fácil é testar o software.
- Usabilidade: diz respeito a quão fácil é para o usuário realizar uma tarefa e o tipo de suporte ao usuário que o sistema provê.

Além das características de qualidade que se aplicam diretamente ao sistema, ditas características de qualidade de produto, Bass, Clements e Kazman (2003) listam outras características relacionadas a metas de negócio, dentre elas: tempo para chegar ao mercado (*time to market*), custo-benefício, tempo de vida projetado para o sistema, mercado alvo, cronograma de implementação e integração com sistemas legados.

Wiegers (2003), por sua vez, agrupa atributos de qualidade do produto em duas categorias principais: atributos importantes para os usuários e atributos importantes para os desenvolvedores. Como atributos importantes para os usuários são apontados os seguintes: disponibilidade, eficiência, flexibilidade, integridade, interoperabilidade, confiabilidade, robustez e usabilidade. Como atributos importantes para os desenvolvedores, são enumerados os seguintes: manutenibilidade, portabilidade, reusabilidade e testabilidade.

Uma vez que não há um consenso sobre quais atributos de qualidade considerar, cada organização deve definir as categorias de requisitos não funcionais a serem consideradas em seus projetos de software. Além disso, essa informação deve ser adicionada à tabela de requisitos não funcionais e, portanto, a Tabela 3.4, quando usada para descrever requisitos não funcionais, deve ter uma coluna adicional para indicar a categoria do requisito não funcional.

Em um mundo ideal, todo sistema deveria exibir os valores máximos para todos os atributos de qualidade. Contudo, como no mundo real isso não é possível, é fundamental definir quais atributos são mais importantes para o sucesso do projeto. Uma abordagem para tratar essa questão consiste em pedir para que diferentes representantes de usuários classifiquem cada atributo em uma escala de 1 (sem importância) a 5 (muito importante). As respostas ajudam o analista a determinar quais atributos são mais importantes. Obviamente, conflitos podem surgir e precisam ser resolvidos (WIEGERS, 2003).

Um aspecto a considerar é que diferentes partes do produto podem requerer diferentes combinações de atributos de qualidade. Assim, é importante também diferenciar características que se aplicam ao produto por inteiro daquelas que são necessárias para certas partes do sistema (WIEGERS, 2003). Para capturar essa diferenciação, é importante introduzir mais uma coluna na tabela de requisitos não funcionais, escopo, podendo assumir dois valores: funcionalidade, quando a característica se aplica apenas a algumas funcionalidades; e sistema, quando a característica se aplica ao sistema como um todo. Quando o escopo de um RNF for de funcionalidade, os requisitos funcionais que precisam incorporar essa característica de qualidade deverão incluir em sua coluna de dependências o identificador desse RNF.

Uma vez priorizados os atributos de qualidade, o analista deve passar, então, a trabalhar com os usuários no sentido de especificar, para cada atributo considerado importante, requisitos mensuráveis e, por conseguinte, testáveis. Se os atributos de qualidade são especificados de maneira não passível de verificação, não é possível dizer posteriormente se eles foram atingidos ou não. Quando apropriado, devem-se indicar escalas ou unidades de medida para cada atributo e os valores mínimo, alvo e máximo. Se não for possível quantificar todos os atributos importantes, deve-se, pelo menos, definir suas prioridades. Pode-se, ainda, perguntar aos usuários o que constituiria um valor inaceitável para cada atributo e definir testes que tentem forçar o sistema a demonstrar tais características (WIEGERS, 2003). É importante destacar, contudo, que essa especificação mais detalhada dos RNFs não deve ser feita no levantamento inicial de requisitos. Ao contrário, ela é considerada como parte da análise de requisitos (corresponde à especificação dos RNFs) e muitas vezes, é até postergada para a fase de projeto (ou transição para a fase de projeto), uma vez que RNFs guardam estreita relação com aspectos tecnológicos, os quais serão tratados na fase de projeto.

Robertson e Robertson (2006) sugerem definir critérios de adequação ou ajuste (*fit criteria*) para permitir quantificar requisitos (tanto funcionais como não funcionais) e associá-los à descrição dos requisitos. À primeira vista, alguns requisitos não funcionais podem parecer difíceis de quantificar. Entretanto, deve ser possível atribuir números a eles. Se não se consegue quantificar e medir um requisito, então é provável que o requisito não seja de fato um requisito. Ele pode ser, por exemplo, vários requisitos descritos em um só.

Seja a seguinte situação. No desenvolvimento de um sistema para uma biblioteca, o usuário coloca o seguinte requisito não funcional: "O sistema deve ser amigável ao usuário". Esse requisito é vago, ambíguo e não passível de ser expresso em números. Como quantificar se o sistema é "amigável ao usuário"? Primeiro, é preciso entender o que o usuário quer dizer com "amigável ao usuário". Significa ser fácil de compreender? Fácil de aprender? Fácil de operar? Atrativo? Clareando a intenção do usuário, é possível sugerir uma escala de medição. Suponha que o usuário diga que ser "amigável ao usuário" significa que os usuários serão capazes de aprender rapidamente a usar o sistema. Uma vez definido que se está falando sobre facilidade de aprender (inteligibilidade), é possível definir como escala de medição o tempo gasto para dominar a execução de certas tarefas. A partir disso, pode-se estabelecer como

critério de aceitação o seguinte: "Novos bibliotecários devem ser capazes de efetuar empréstimos após a quarta tentativa de realizar essa tarefa usando o sistema".

Determinar critérios de aceitação ajuda a clarear um requisito. Ao se estabelecer uma escala de medição e os valores aceitáveis, o requisito é transformado de uma intenção vaga, e até certo ponto ambígua, em um requisito mensurável e bem formado. Estabelecida uma escala, pode-se perguntar ao usuário o que ele considera uma falha em atender ao requisito, de modo a definir o critério de aceitação. Contudo, pode ser difícil, senão impossível, obter um requisito completo e mensurável em primeira instância (ROBERTSON; ROBERTSON, 2006). Assim, na descrição de requisitos de cliente é suficiente capturar a intenção e depois, na especificação de requisitos de sistema, transformar essa intenção em um requisito mensurável, adicionando a ele um critério de aceitação. É muito comum que, neste processo, um requisito não funcional de usuário dê origem a vários requisitos não funcionais de sistema.

3.5 - Regras de Negócio

Toda organização opera de acordo com um extenso conjunto de políticas corporativas, leis, padrões industriais e regulamentações governamentais. Tais princípios de controle são coletivamente designados por regras de negócio. Uma regra de negócio é uma declaração que define ou restringe algum aspecto do negócio, com o propósito de estabelecer sua estrutura ou controlar ou influenciar o comportamento do negócio (WIEGERS, 2003).

Sistemas de informação tipicamente precisam fazer cumprir as regras de negócio. Ao contrário dos requisitos funcionais e não funcionais, a maioria das regras de negócio origina-se fora do contexto de um sistema específico. Assim, as regras a serem tratadas pelo sistema precisam ser identificadas, documentadas e associadas aos requisitos do sistema em questão (WIEGERS, 2003).

Wiegers identifica cinco tipos principais de regras de negócio, cada um deles apresentando uma forma típica de ser escrito:

- Fatos ou invariantes: declarações que são verdade sobre o negócio. Geralmente descrevem associações ou relacionamentos entre importantes termos do negócio. Ex.: Todo pedido tem uma taxa de remessa.
- Restrições: como o próprio nome indica, restringem as ações que o sistema ou seus usuários podem realizar. Algumas palavras ou frases sugerem a descrição de uma restrição, tais como *deve*, *não deve*, *não pode* e *somente*. Ex.: Um aluno só pode tomar emprestado, concomitantemente, até três livros.
- Ativadores de Ações: são regras que disparam alguma ação sob condições específicas. Uma declaração na forma "Se <alguma condição é verdadeira ou algum evento ocorre>, então <algo acontece>" é indicada para descrever ativadores de ações. Ex.: Se a data para retirada do livro é ultrapassada e o livro não é retirado, então a reserva é cancelada. Quando as condições que levam às ações são uma complexa combinação de múltiplas condições individuais, então o uso de tabelas de decisão ou árvores de decisão é indicado. As figuras 3.4 e 3.5 ilustram uma mesma regra de ativação de ações descrita por meio de uma árvore de decisão e de uma tabela de decisão, respectivamente.

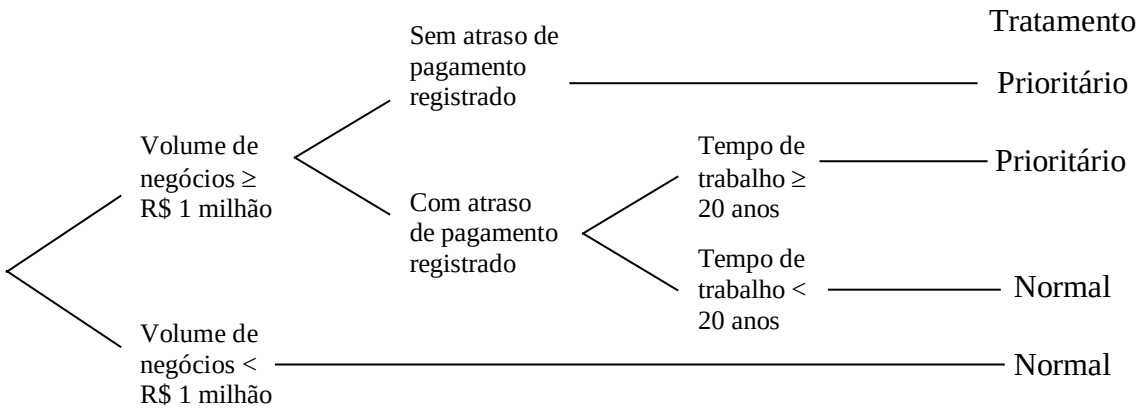

Figura 3.4 – Exemplo de Árvore de Decisão.

Tratamento de Clientes				
Volume de Negócios ≥ R$ 1 milhão?	S	S	S	N
Atraso de pagamento registrado?	N	S	S	-
Tempo de trabalho ≥ 20 anos?	-	S	N	-
Tratamento Prioritário	X	X		
Tratamento Normal			X	X

Figura 3.5 – Exemplo de Tabela de Decisão.

- Inferências: são regras que derivam novos fatos a partir de outros fatos ou cálculos. São normalmente escritas no padrão "se / então", como as regras ativadoras de ação, mas a cláusula então implica um fato ou nova informação e não uma ação a ser tomada. Ex.: Se o usuário não devolve um livro dentro do prazo estabelecido, então ele torna-se um usuário inadimplente.
- Computações: são regras de negócio que definem cálculos a serem realizados usando fórmulas matemáticas ou algoritmos específicos. Podem ser expressas como fórmulas matemáticas, descrição textual, tabelas etc. Ex.: Multa = Valor de Locação * Número de Dias de Atraso.

Ao contrário de requisitos funcionais e não funcionais, regras de negócio não são passíveis de serem capturadas por meio de perguntas simples e diretas, tal como "Quais são suas regras de negócio?" Regras de negócio emergem durante a discussão de requisitos, sobretudo quando se procura entender a base lógica por detrás de requisitos e restrições apontados pelos interessados (WIEGERS, 2003). Assim, não se deve pensar que será possível levantar muitas regras de negócio em um levantamento preliminar de requisitos. Pelo contrário, as regras de negócio vão surgir principalmente durante o levantamento detalhado dos requisitos. Wiegers (2003) aponta diversas potenciais origens para regras de negócio e sugere tipos de questões que o analista pode fazer para tentar capturar regras advindas dessas origens:

- Políticas: Por que é necessário fazer isso desse jeito?
- Regulamentações: O que o governo requer?
- Fórmulas: Como este valor é calculado?
- Modelos de Dados: Como essas entidades de dados estão relacionadas?
- Ciclo de Vida de Objetos: O que causa uma mudança no estado desse objeto?
- Decisões de Atores: O que o usuário pode fazer a seguir?
- Decisões de Sistema: Como o sistema sabe o que fazer a seguir?
- Eventos: O que pode (e não pode) acontecer?

Regras de negócio normalmente têm estreita relação com requisitos funcionais. Uma regra de negócio pode ser tratada no contexto de certa funcionalidade. Assim, a regra de negócio deve ser associada ao requisito funcional. Há casos em que uma regra de negócio conduz a um requisito funcional para fazer cumprir a regra. Neste caso, a regra de negócio é considerada a origem do requisito funcional (WIEGERS, 2003).

É importante destacar a importância das regras de restrição obtidas a partir de modelos conceituais estruturais, ditas *restrições de integridade*. Elas complementam as informações de um modelo deste tipo e capturam restrições relativas a relacionamentos entre elementos de um modelo que normalmente não são passíveis de serem capturadas pelas notações gráficas utilizadas na elaboração de modelos conceituais estruturais. Tais regras devem ser documentadas junto ao modelo conceitual estrutural do sistema. Seja o exemplo do modelo conceitual estrutural da Figura 3.6. Esse fragmento de modelo indica que: (i) um aluno cursa um curso; (ii) um aluno pode se matricular em nenhuma ou várias turmas; (iii) um curso possui um conjunto de disciplinas em sua matriz curricular; (iv) uma turma é de uma disciplina específica. Contudo, nada diz sobre restrições entre o estabelecimento dessas várias relações. Suponha que o negócio indique que a seguinte restrição deve ser considerada: Um aluno só pode ser matricular em turmas de disciplinas que compõe a grade curricular do curso que esse aluno cursa. Essa restrição tem de ser escrita para complementar o modelo. Sugere-se utilizar "RI" como o prefixo para o identificador único de restrições de integridade.

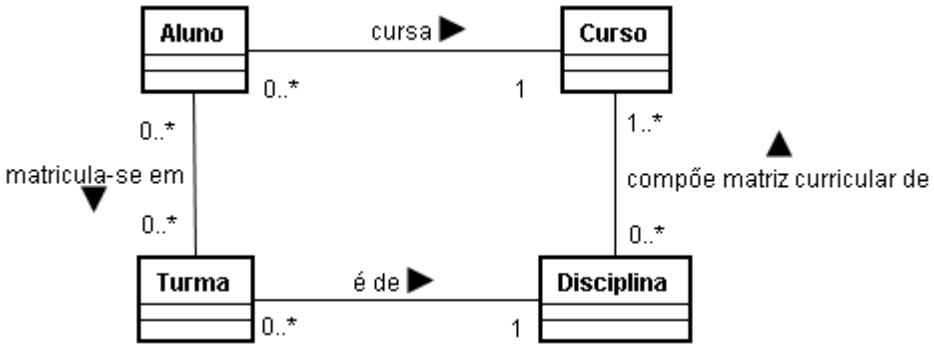

Figura 3.6 – Exemplo de Fragmento de Modelo de Dados com Restrição de Integridade.

Outro tipo de restrição importante são as regras que impõem restrições sobre funcionalidades de inserção, atualização ou exclusão de dados, devido a relacionamentos existentes entre as entidades. Voltando ao exemplo da Figura 3.6, a exclusão de disciplinas não deve ser livre, uma vez que turmas são existencialmente dependentes de disciplinas. Assim, a

seguinte regra deve ser considerada: Ao excluir uma disciplina, devem-se excluir todas as turmas a ela relacionadas. Essas regras são denominadas neste texto como *restrições de processamento* e, uma vez que dizem respeito a funcionalidades, devem ser documentadas junto com a descrição do caso de uso que detalha a respectiva funcionalidade. Neste caso, não há necessidade de se identificar unicamente a restrição de processamento usando identificadores, uma vez que ela será documentada diretamente com o caso de uso relacionado.

3.6 - Suposições

O comportamento do sistema ou os resultados esperados de sua utilização podem ser pautados em suposições. Uma suposição descreve um estado de coisas no ambiente de interesse do sistema que os interessados acreditam ser verdadeiro. Em outras palavras, uma suposição é uma crença de que uma situação específica acontece no ambiente de interesse. Algumas vezes, representar tais situações é útil, pois elas precisam ser consideradas na solução de um problema específico. Mais do que isso, as suposições podem ser imprescindíveis para dizer se um sistema atinge ou não seus objetivos (NEGRI et al., 2017). P.ex., em um sistema de biblioteca, quando um usuário tem uma reserva, ele deve ser notificado quando um exemplar do livro reservado estiver disponível para atender sua reserva. Visando satisfazer o objetivo de comunicar ao usuário a possibilidade de atendimento de sua reserva, uma funcionalidade (requisito funcional) é desenvolvida para enviar um e-mail para o usuário, avisando-o da disponibilidade. Essa funcionalidade sozinha, contudo, não garante que o objetivo de "comunicar o usuário" é atingido. Para que esse objetivo seja, de fato, atingido, há suposições a considerar, tais como o e-mail registrado no sistema está atualizado, a mensagem será entregue e o usuário vai acessar e ler o e-mail.

Representar todas as suposições relacionadas aos requisitos não é uma prática comum. E mais: pode não ser necessário, uma vez que muitas delas são óbvias. Contudo, explicitar suposições pode ser necessário quando as suposições não forem óbvias ou quando o engenheiro de requisitos quiser enfatizá-las (NEGRI et al., 2017). Assim, quando pertinente, tabelas descrevendo suposições podem ser adicionadas, tipicamente, a Documentos de Especificação de Requisitos.

Referências do Capítulo

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

BASS, L., CLEMENTS, P., KAZMAN, R., *Software Architecture in Practice*, Second edition, Addison Wesley, 2003.

CARDOSO, E.C.S., Uma Comparação entre Requisitos de Sistema Gerados por Técnicas de Modelagem de Processos com Requisitos de Sistema Gerados por Técnicas Convencionais de Engenharia de Requisitos, Projeto de Graduação (Engenharia de Computação), Universidade Federal do Espírito Santo, 2007.

CARDOSO, E., ALMEIDA, J.P.A., GUIZZARDI, G., "Uma Experiência com Engenharia de Requisitos baseada em Modelos de Processos". In: Proceedings of the XI Iberoamerican

Workshop on Requirements Engineering and Software Environments (IDEAS 2008), Recife, 2008.

CARVALHO, E.A., Engenharia de Processos de Negócio e a Engenharia de Requisitos: Análise e Comparações de Abordagens e Métodos de Elicitação de Requisitos de Sistemas Orientada por Processos de Negócio, Dissertação de Mestrado, Programa de Pós-Graduação em Engenharia de Produção, COPPE/UFRJ, Rio de Janeiro, 2009.

DAVENPORT, T. H., *Mission Critical: realizing the promise of enterprise systems.* 1st edition. Boston: Harvard Business School Press, 2000.

FRYE, D. W., GULLEDGE, T. R., "End-to-end Business Process Scenarios". *Industrial, Management & Data Systems*, v. 107, n. 6, pp.749–761, 2007.

ISO/IEC 25010, System and Software Engineering - Systems and software Quality Requirements and Evaluation (SQuaRE) - System and software quality models, 2011.

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

KNIGHT, D. M. Elicitação de Requisitos de Software a partir do Modelo de Negócio. Dissertação (Mestrado em Informática). Núcleo de Computação Eletrônica (NCE), Universidade Federal do Rio de Janeiro. Rio de Janeiro, 2004

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

NEGRI, P.P., SOUZA, V.E.S., LEAL, A.LC., FALBO, R.A., GUIZZARDI, G., Towards an Ontology of Goal-Oriented Requirements, XX Ibero-American Conference on Software Engineering - CIbSE'2017, Buenos Aires, Argentina, 2017.

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.

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

WAZLAWICK, R.S., *Análise e Projeto de Sistemas de Informação Orientados a Objetos*, Elsevier, 2004.

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