UNIVERSIDADE FEDERAL DE PERNAMBUCO
CENTRO DE INFORMÁTICA

GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

ESTUDO COMPARATIVO DE MÉTODOS DE GERAÇÃO DE CASOS DE USO

Davi de França Carneiro

Trabalho de Graduação

Recife
SETEMBRO DE 2013

UNIVERSIDADE FEDERAL DE PERNAMBUCO
CENTRO DE INFORMÁTICA

Davi de França Carneiro

ESTUDO COMPARATIVO DE MÉTODOS DE GERAÇÃO DE CASOS DE USO

Trabalho apresentado ao Programa de GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO do CENTRO DE INFORMÁTICA da UNIVERSIDADE FEDERAL DE PERNAMBUCO como requisito parcial para obtenção do grau de Bacharel em CIÊNCIA DA COMPUTAÇÃO.

Orientadora: *Profª Drª Carla Taciana Lima Lourenço Silva Schuenemann*

Recife
SETEMBRO DE 2013

É o momento de dedicar este trabalho aos meus pais
Mirian e Marcos, meus exemplos de vida.

Agradecimentos

Meus mais sinceros e eternos agradecimentos

Aos meus pais, Mirian e Marcos, por tudo

À família, pelo incentivo e presença em situações de alegrias e tristezas

À minha orientadora Carla, por muito ter me ensinado e ajudado nesta empreitada

A Dyogo Braga, pela força que tem me dado e pela confiança em todos os momentos

Aos amigos, pelo apoio, compreensão e momentos de risadas e descontração

Aos colegas, pela companhia e cumplicidade

À vida, pelo prazer de ser vivida

E a Deus, por ter me proporcionado todos esse motivos para agradecer.

Resumo

Para os estudiosos da área de engenharia de requisitos, é consensual a necessidade de se obter/identificar métodos capazes de gerar, de maneira sistemática, especificações de casos de uso mais corretas e completas, possibilitando que os requisitos do sistema estejam o mais alinhado possível às metas da organização. Na literatura, pesquisas têm sido realizadas, utilizando diferentes abordagens para a engenharia de requisitos, como por exemplo a orientada a objetivos e a baseada em processos de negócio. No primeiro caso, o uso do *framework* i* tem se destacado para raciocinar e modelar o ambiente organizacional. O i* centra-se nos atores organizacionais, que dependem uns dos outros para que os objetivos sejam alcançados, tarefas sejam executadas e recursos sejam fornecidos. No segundo caso, o uso da notação BPMN tem ganhado espaço na modelagem dos processos de negócio, tornando a captura do funcionamento complexo das organizações uma tarefa mais simples, compreensível pelos *stakeholders* e condizente com a realidade do negócio. O objetivo deste trabalho é, portanto, realizar um estudo comparativo do método de Santander (que utiliza a abordagem orientada a objetivos) e de De la Vara (que utiliza a abordagem baseada em processos de negócio) para gerar especificações de cenários de casos de uso. O uso destes métodos possibilita aos engenheiros de requisitos, bem como aos demais envolvidos, a especificação de requisitos de software mais correta, completa e alinhada às metas da organização. Com este estudo comparativo, pode-se identificar o método considerado mais adequado, de acordo com um conjunto de métricas e questões que avaliam cada método.

Palavras-chave: *Framework* i*, Engenharia de Requisitos Orientada a Objetivos, BPMN, Engenharia de Requisitos Baseada em Processos de Negócio, Caso de Uso, ETD.

Abstract

For researchers in the area of requirements engineering, it is a consensus the need for obtaining/identifying methods able to systematically generate complete and correct use cases, enabling system requirements to be as aligned as possible to the goals of the organization. In the literature, researches have been done using different approaches for the requirements engineering, such as the goal-oriented and business process-based approaches. In the first case, the use of the i* framework has been highlighted to rationale and model the organizational environment. The i* focuses on the organizational actors who depend on each other, so that goals are achieved, tasks are performed and resources are provided. In the second case, the use of the BPMN notation has gained space on the business process modeling, making the capture of complex functioning of organizations a simpler task, easier to understand by the stakeholders and consistent with the business reality. The purpose of this work is, therefore, to conduct a comparative study between the Santander's method (which uses the goal-oriented approach) and De la Vara's method (which uses the approach based on business processes) to generate specifications of use cases scenarios. Using these methods enables requirements engineers, as well as others involved in software requirements, to specify software requirements more correctly, completely and aligned with the organization's goals. Furthermore, the comparative study provided by this work helps to identify the method considered most suitable, according to a suite of metrics and questions that evaluate each method.

Keywords: i* Framework, Goal-Oriented Requirements Engineering, BPMN, Business Process-Based Requirements Engineering, Use Case, ETD.

Sumário

Lista de Figuras

Lista de Tabelas

CAPÍTULO 1

Introdução

"O senso comum é o menos comum dos sensos"

Anônimo

Este capítulo apresenta os aspectos introdutórios desta monografia, organizados em quatro seções. Inicialmente, é descrito o contexto em que se insere este trabalho. Posteriormente, é apresentada uma seção com as motivações que regem a elaboração deste estudo. Em seguida, são relatados os objetivos propostos e, por fim, dedica-se uma seção que apresenta a estrutura do documento.

1.1. Contexto

A engenharia de software é uma área da computação que engloba atividades de especificação, implementação e manutenção de sistemas de software [3]. No contexto da engenharia de software, temos a engenharia de requisitos, que abrange a fase inicial de desenvolvimento do sistema. A engenharia de requisitos se utiliza de técnicas de levantamento, documentação e análise de requisitos, buscando garantir a completude e consistência do produto final, ou seja, do software [5].

Os requisitos de software, que correspondem às características que o software deve possuir ou atender, podem ser descobertos ou especificados de diversas formas, como por exemplo, através da abordagem orientada a objetivos. Esta abordagem foca nas metas do usuário, isto é, na expectativa do usuário com relação ao que o sistema deve fazer ou como ele deve se comportar. Assim, esta abordagem evidencia o motivo principal pelo qual o sistema deve ser desenvolvido [1].

Como representante dessa, pode-se citar o *framework* i*, que foi desenvolvido para raciocinar e modelar o ambiente organizacional e seus sistemas de informação. O i* centra-se no ator intencional, isto é, nos atores organizacionais vistos como possuidores de propriedades intencionais, tais como objetivos, crenças, habilidades e compromissos. Assim, estes atores dependem uns dos outros para que os objetivos sejam alcançados, tarefas sejam executadas e recursos sejam fornecidos [9].

Por outro lado, o uso da notação BPMN (*Business Process Model and Notation*), bastante utilizada e difundida para a modelagem de processos organizacionais, tem sido uma "ferramenta" muito poderosa para a obtenção de informações do funcionamento da organização e de todos os elementos envolvidos nos seus processos. Sendo assim, uma abordagem baseada em processos de negócio tem sido usada para obter especificações de casos de uso e requisitos do sistema [2][7].

No contexto de requisitos de software, há os chamados requisitos funcionais que correspondem às funcionalidades do sistema. Tais requisitos estão associados aos casos de uso, os quais descrevem narrativamente uma sequência de eventos gerados a partir da ação de um ator (agente externo) ao executar alguma tarefa no sistema [6]. Os casos de uso, portanto, mapeiam o escopo do sistema, facilitam a comunicação com o usuário do sistema e possibilitam melhor gerenciamento do projeto de desenvolvimento.

Um caso de uso pode ser visto como um conjunto de cenários, onde cada cenário contém uma sequência de passos que representa uma interação entre o usuário e o sistema. É possível imaginar, por exemplo, que uma tarefa seja executada com diferentes caminhos. Assim são os cenários, eles apresentam distintas possibilidades na execução do caso de uso, que levem ao sucesso ou não [8].

Na próxima seção, são apresentadas as motivações para a elaboração deste trabalho, considerando as duas abordagens anteriormente descritas.

1.2. Motivações

Diante das diferentes abordagens da engenharia de requisitos, mencionadas na seção anterior, alguns métodos surgiram com o intuito de gerar, de maneira sistemática, especificações de casos de uso. A saber, o método de Santander [4] usa o i* e o método de De la Vara [7] usa o BPMN.

Com isto, percebe-se a necessidade de auxiliar o engenheiro de requisitos na indicação do método que mais se adéque às suas necessidades, no que diz respeito à obtenção de especificação de casos de uso de um sistema e seus respectivos cenários. Neste sentido, o presente trabalho destina-se a realizar um estudo comparativo entre os métodos de Santander e De la Vara, explicitando o funcionamento de cada um, definindo métricas de comparação e concretizando resultados através da avaliação de cada método.

A próxima seção apresenta os objetivos que se deseja alcançar com este trabalho.

1.3. Objetivos Propostos

O objetivo principal deste trabalho é realizar um estudo comparativo entre os métodos de Santander e De la Vara. Especificamente, podemos descrever os objetivos conforme segue:

- Levantar os principais conceitos e fundamentos, e entender o funcionamento dos métodos;
- Identificar métricas de comparação dos métodos, segundo os objetivos que se deseja avaliar (corretude, completude, complexidade do método e alinhamento dos requisitos com as metas organizacionais);
- Aplicar as métricas identificadas para a avaliação dos métodos, no contexto do exemplo agendador de reuniões;
- Apresentar os resultados obtidos, para que estes sirvam de guia para auxiliar um engenheiro de requisitos, bem como os demais envolvidos no projeto, a identificar o método que melhor se adéque às suas necessidades para especificação dos casos de uso do sistema.

A próxima seção apresenta a estrutura deste documento, apontando o conteúdo que será abordado ao longo deste trabalho.

1.4. Estrutura do Documento

Este trabalho se inicia com o capítulo 2, que mostrará ao leitor os aspectos teóricos essenciais para um bom entendimento de todo o documento. Desta forma, são descritos os principais conceitos envolvidos na engenharia de requisitos (níveis e tipos de requisitos, processo da engenharia de requisitos, casos de uso e cenários) e, também, as abordagens da engenharia de requisitos (orientada a objetivos, com destaque à modelagem utilizando o i*, e baseada em processos de negócio, utilizando a notação BPMN).

No capítulo 3, a monografia apresentará o primeiro objeto do estudo comparativo apresentado neste trabalho: o método de Santander. Assim, é descrita uma visão geral sobre a integração entre modelos orientados a objetivos e casos de uso. Em seguida, é apresentado o método de Santander com suas diretrizes, cuja aplicação é contextualizada no exemplo agendador de reuniões.

No capítulo 4, será apresentado o segundo objeto do estudo comparativo realizado neste trabalho: o método de De la Vara. Inicialmente, é mostrada uma visão geral sobre *As-Is* BPDs (*Business Process Diagrams*) e *To-Be* BPDs. Posteriormente, são introduzidos os conceitos de *Labelled* BPD e *Enriched* BPD. Por fim, é apresentado o método de De la Vara com suas diretrizes, cuja aplicação é contextualizada no exemplo agendador de reuniões.

No capítulo 5 será apresentado o estudo comparativo dos métodos de Santander e De la Vara. Inicialmente, é descrita uma breve motivação para a realização da comparação. Em seguida são definidas as métricas de comparação, baseando-se no conceito GQM (*Goal Question Metric*) [10]. Por fim, é feita a avaliação dos métodos, para a identificação dos resultados, aplicando-se as métricas definidas.

No capítulo 6, serão apresentadas as conclusões deste trabalho, ressaltando as contribuições e limitações encontradas, bem como as perspectivas futuras. Por fim, o capítulo descreve algumas considerações finais advindas da elaboração desta monografia.

CAPÍTULO 2

Fundamentação Teórica

"Sem a teoria, a prática nada mais é que a rotina dada pelo hábito"

Louis Pasteur

Este capítulo apresenta os aspectos teóricos essenciais para o bom entendimento deste trabalho. É fundamental entender os níveis e tipos dos requisitos e os processos envolvidos na engenharia de requisitos de modo que seja possível compreender o contexto em que ela se insere no relacionamento sistema - organização. Além disso, neste capítulo é mostrada a abordagem de engenharia de requisitos orientada a objetivos chamada de *framework* i*. Posteriormente, são exibidos os conceitos fundamentais sobre a engenharia de requisitos baseada na modelagem de processos de negócio utilizando a notação BPMN e, por fim, o capítulo se encerra descrevendo o conceito de casos de uso, bem como a definição de cenários dentro deste contexto.

2.1. Engenharia de Requisitos - Visão Geral

A maioria das organizações utilizam algum sistema de informação (SI) para realizar o gerenciamento, armazenamento e/ou provimento de informações. Esse tipo de sistema faz parte do nicho da organização, apoiando a realização de tarefas tipicamente pertencentes aos seus processos de negócio. Desse modo, a necessidade de se realizar uma especificação de requisitos mais correta, completa e alinhada às expectativas organizacionais tem sido pauta de muitos estudos da área de engenharia de requisitos [4][7]. Estes estudos lançam mão de diversas perspectivas e modelagens das interações entres os envolvidos na organização bem como dos seus processos de negócio, em busca de uma melhor especificação de requisitos.

Enquanto a engenharia de software é uma área ampla da computação responsável pela especificação, implementação e manutenção de sistemas de software [3], a engenharia de requisitos abrange principalmente a fase inicial do desenvolvimento do sistema. Assim, ela utiliza técnicas de levantamento, documentação e análise de requisitos, buscando garantir a completude e consistência do produto final, ou seja, do software [5].

A engenharia de requisitos é responsável por manter a concordância entre clientes e desenvolvedores, registrar e realizar o acompanhamento contínuo dos requisitos durante toda a fase de desenvolvimento, determinar quais serão as fronteiras do sistema e também fornece uma base para que seja possível estimar os custos e prazos de desenvolvimento do sistema [11].

Realizar o levantamento de requisitos de um SI não é uma tarefa fácil, uma vez que muitas dificuldades podem existir durante essa etapa. Dentre essas dificuldades, podemos citar principalmente a deficiência de comunicação e de sincronia de pensamentos entre o engenheiro de requisitos, o qual geralmente possui uma gama de conhecimentos teóricos e técnicos e os *stakeholders*[1], os quais possuem o conhecimento prático do negócio e do domínio envolvido [11].

Portanto, utilizar instrumentos capazes de reduzir essa dissonância é fundamental para o sucesso da especificação de requisitos de um SI. De fato, um bom entendimento do ambiente organizacional pode ser alcançado utilizando técnicas de modelagem e análise que permitam alinhar os requisitos do sistema com as expectativas dos *stakeholders*. Isto ajudaria a deixar tanto o engenheiro de requisitos quanto as demais partes envolvidas confortáveis e confiáveis na hora de definir o que de fato irá compor o sistema, evitando inconsistência de informações [9].

A seguir, serão detalhados os níveis e tipos dos requisitos de software. Este entendimento é importante para identificar se determinado requisito se enquadra em nível de usuário ou de sistema e se corresponde a funcionalidades que o sistema deve possuir, a quesitos de qualidade ou restrições, ou a regras de negócio.

2.1.1. Níveis e Tipos dos Requisitos

Os requisitos de software correspondem às características que o software deve possuir ou atender para que o sistema cumpra seu objetivo. Segundo [12], os requisitos se classificam em dois níveis (usuário e sistema) e em três tipos (funcionais, não funcionais e de domínio).

Os requisitos de usuário são aqueles que contêm declarações em linguagem natural e que são facilmente compreensíveis com a utilização de diagramas. Destinam-se às pessoas envolvidas com o uso e com a aquisição do sistema, como gerentes de clientes, usuários finais

[1] *Stakeholders* é o termo dado às partes interessadas e que são elementos essenciais ao planejamento estratégico dos negócios.

do sistema, engenheiros do cliente, gerentes do fornecedor e arquitetos do sistema. Esses requisitos definem o que o sistema deve fazer, considerando as restrições sob as quais ele deve operar [12].

Os requisitos de sistema são estruturados em um documento que contém, em detalhes, os serviços/funções que são fornecidos pelo sistema. Esse documento é escrito como um contrato entre as partes envolvidas (cliente e contratante). É importante salientar que nesse nível de requisitos, deve-se definir o que o sistema deve fazer, e não como ele deve ser feito em nível de implementação. Esses requisitos destinam-se, por exemplo, a usuários finais do sistema, engenheiros do cliente, arquitetos do sistema e desenvolvedores de software [12].

Com relação aos tipos, os requisitos funcionais contêm declarações de funções fornecidas pelo sistema, especificando como o mesmo deve se comportar dadas entradas específicas e em determinadas situações [12].

Os requisitos não-funcionais, os quais podem ser de produto (relacionados ao comportamento do produto), organizacionais (decorrentes da política e cultura organizacional) ou externos (fora do sistema ou das etapas de desenvolvimento), expressam quesitos de qualidade ou restrições sobre as quais as funções fornecidas pelo sistema devem trabalhar (ex.: restrições de tempo, processos de desenvolvimento e padrões) [12].

Muitos requisitos não-funcionais surgem de acordo com a necessidade de quem vai usar o sistema e, obviamente, em função do quanto o cliente está disposto a desembolsar (orçamento) para garantir tais requisitos. É importante notar que embora os requisitos não-funcionais não estejam diretamente ligados às funcionalidades, o não cumprimento de muitos desses requisitos pode "jogar no lixo" todo o sistema (ex.: um requisito de segurança numa transação bancária online).

Os requisitos de domínio, muitas vezes não citados por muitos autores, representam requisitos específicos do domínio da aplicação, isto é, as regras de negócio que devem ser respeitadas pelo sistema [12].

A seguir, será introduzido o conceito de processo da engenharia de requisitos. É importante entender as etapas que compõem esse processo, pois a devida execução dessas etapas levam ao sucesso da especificação de requisitos.

2.1.2. Processo da Engenharia de Requisitos

Um processo nada mais é que um conjunto de atividades que são executadas de maneira estruturada. O processo de engenharia de requisitos visa extrair, validar e gerenciar um documento de requisitos. Conforme [12], o processo da engenharia de requisitos (ver figura 2.1) é composto por várias etapas bem definidas:

- Estudo de viabilidade: decide se vale a pena gastar tempo, esforço e dinheiro com o projeto. Resulta em um documento de viabilidade;
- Elicitação e análise de requisitos: envolve o relacionamento entre pessoal técnico e *stakeholders* para entender o domínio da aplicação e conhecer os serviços, funções e restrições para o sistema. Resulta em modelos de sistema;
- Especificação de requisitos: após o processo de elicitação e análise, gera-se uma especificação com os requisitos de usuário e de sistema;
- Validação de requisitos: verifica se tudo está conforme desejado pelo cliente. Fundamental para a elaboração do documento de requisitos.

É interessante ressaltar que o gerenciamento de mudanças em qualquer ponto do processo é fundamental, uma vez que na maioria das vezes os clientes podem solicitar mudanças no escopo do projeto com a inclusão, remoção ou alteração de requisitos.

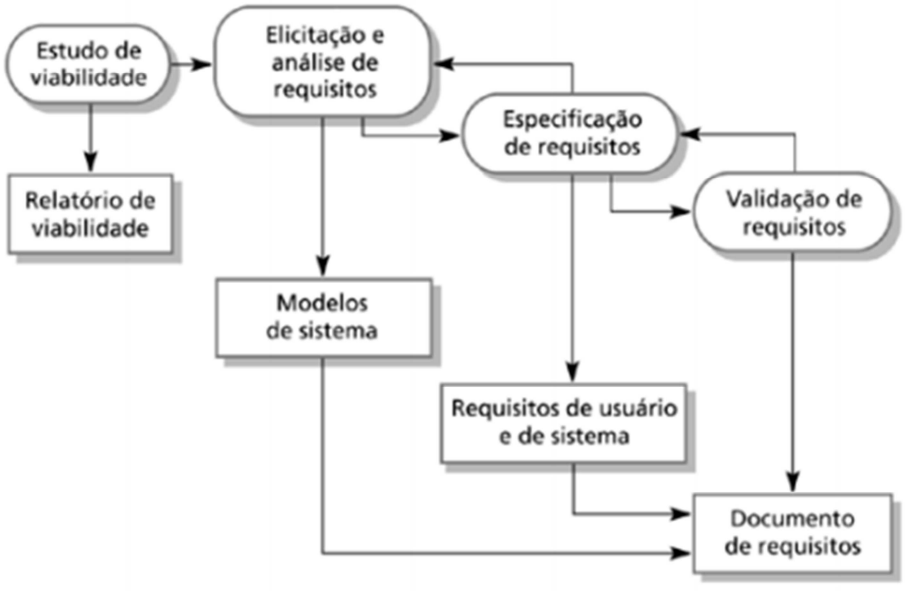

Figura 2.1 Processo de engenharia de requisitos [12].

No momento da descoberta de requisitos, algumas técnicas são bastante conhecidas e utilizadas. Dentre elas, pode-se citar [13]:

- Perguntar: consiste em saber identificar a pessoa apropriada que vai contribuir com a extração dos requisitos;
- Observar e inferir: consiste em observar como os usuários trabalham para entender suas necessidades. Em geral essa técnica demanda bastante tempo;
- Discutir e formular: consiste em discutir com os usuários informalmente, tentando absorver seus conceitos e necessidades para formular uma versão comum e preliminar dos requisitos;
- Negociar: consiste em analisar os requisitos levantados, para negociar quais serão levados a cabo, quais serão eliminados e quais necessitam de modificações;
- Identificar problemas: consiste em identificar problemas durante a execução das atividades rotineiras dos usuários visando extrair requisitos que possam melhorar o produto;
- Supor: consiste da utilização da intuição do engenheiro de requisitos para definir os requisitos do sistema quando não se tem usuários que possam colaborar com o processo;
- Técnicas informais: consiste da utilização de elementos informais que precisam de comunicação direta e estruturada com os *stakeholders*. Exemplos: entrevistas, questionários, *brainstorming*, entre outros;
- Técnicas formais: consiste da utilização formal de modelos no contexto do problema a ser analisado de modo a extrair requisitos para o sistema. Além disso, protótipos do software a ser desenvolvido também podem ser usados para descobrir requisitos.

Na próxima seção, é apresentado o conceito de engenharia de requisitos orientada a objetivos. Será detalhado como os objetivos organizacionais podem ser modelados para sua utilização na especificação de requisitos.

2.2. Engenharia de Requisitos Orientada a Objetivos

Nos últimos anos, muitos estudiosos têm dedicado seu tempo na busca de melhores técnicas de engenharia de esquisitos, de modo que o resultado obtido possa incluir de maneira correta e completa as funcionalidades do sistema e um maior alinhamento às metas organizacionais.

Assim, o sistema especificado deve atender às necessidades da organização e apoiar na execução dos seus processos de negócio, inclusive levando em consideração o inter-relacionamento entre os participantes desse processo.

Os requisitos de software podem ser descobertos ou especificados de diversas formas, utilizando as técnicas citadas anteriormente. Dentre as técnicas formais, encontra-se a engenharia de requisitos orientada a objetivos que foca nas metas do usuário, isto é, na expectativa do usuário com relação ao que o sistema deve fazer ou como ele deve se comportar. Assim, esta abordagem evidencia o motivo principal pelo qual o sistema deve ser desenvolvido [1].

As abordagens orientadas a objetivos visam expressar os "porquês" atrelados com a criação do sistema [4]. Dessa forma, os requisitos que compõem o sistema são bem fundamentados e justificados. É importante ressaltar que os requisitos levantados para um sistema são tão bons quanto o grau de satisfação dos objetivos propostos com o projeto [14].

A ideia é que, com base nos objetivos bem compreendidos tanto pelos *stakeholders* quanto pela equipe técnica, todos os envolvidos possam contribuir mais confortavelmente no processo inicial da engenharia de requisitos, fase na qual geralmente as maiores dificuldades são encontradas, como a dificuldade de comunicação e compreensão da linguagem de especificação de requisitos adotada [11].

Uma grande vantagem, além da captura dos objetivos, é que a abordagem emprega modelos que exprimem visualmente os relacionamentos de dependência entre os atores do ambiente organizacional, bem como todos os elementos fundamentais envolvidos nessa relação. Há uma perspectiva de colaboração entre os atores para que os objetivos sejam alcançados [9].

Por se tratar de uma modelagem organizacional, todos os tipos de requisitos mencionados na seção 2.1.1 são modelados, o que permite maior englobamento dos requisitos.

Como representante dessa abordagem orientada a objetivos, pode-se citar o *framework* i*, que foi desenvolvido para raciocinar e modelar processos de negócio e seus sistemas de informação. O i* centra-se no ator intencional, isto é, nos atores organizacionais vistos como possuidores de propriedades intencionais, tais como objetivos, crenças, habilidades e compromissos. Os atores dependem uns dos outros para que os objetivos sejam alcançados, tarefas sejam executadas e recursos sejam fornecidos [9].

A seguir, é apresentado como é feita a modelagem organizacional utilizando o i*, ressaltando os dois modelos que compõem este *framework* (Modelo de Dependências Estratégicas e Modelo de Razões Estratégicas).

2.2.1. Modelagem com o *Framework* i*

O *framework* i* foi desenvolvido visando ampliar as dimensões de representação e modelagem de aspectos organizacionais em seu processo de negócio [9]. Os pontos chaves do i* consistem em retratar intencionalidades e motivações que levam à execução de processos por atores dentro do ambiente da organização. Segundo [9], o conceito de ator do ambiente organizacional é extremamente importante porque é ele quem possui propriedades intencionais, como objetivos, executa tarefas e aprovisiona recursos.

O i*, diferentemente de muitas visões tradicionais baseadas em fluxos de atividades e informações, procura introduzir os relacionamentos sociais presentes no dia a dia da organização. Com isso, a análise social proporcionada por essa abordagem passa a se preocupar com as metas desejadas pelos atores, como eles pretendem atingir tais metas e de que recursos dependem [15].

O i* permite uma maior compreensão da organização como um todo e possibilita que sistemas de software sejam incluídos no processo de negócio durante a modelagem. Além disso, torna-se perceptível como propostas alternativas de sistemas podem impactar o ambiente organizacional, inclusive como se tornariam os relacionamentos entre atores com essa modificação no processo de negócio. Dessa forma, tal *framework* auxilia os engenheiros de requisitos a identificarem os possíveis impactos que a inserção de um sistema de software poderia ocasionar tanto na organização quanto nos seus clientes, o que não é uma tarefa tão simples assim [9].

Como a finalidade é determinar as razões ou os "porquês" associados aos requisitos de software, a modelagem utilizando esta técnica inclui dois tipos de modelos, que juntos são fundamentais para alcançar os propósitos por ela estabelecidos. Esses modelos são o Modelo de Dependências Estratégicas (SD) e o Modelo de Razões Estratégicas (SR), os quais são mostrados a seguir.

Modelo de Dependências Estratégicas - SD

O modelo de dependências estratégicas é estruturado através de relações de dependências entre elementos que representam os atores organizacionais [9]. O ator é a entidade responsável por executar determinadas ações visando alcançar os objetivos especificados no contexto da organização. Entende-se por dependência a necessidade que um ator possui em relação a outro para alcançar um objetivo, um objetivo-soft, realizar uma tarefa ou dispor de um recurso.

Conforme mencionado em [9], o modelo SD contém informações de mais alto nível, sem se preocupar com os detalhes e refinamentos dos seus elementos. Por isso, o SD apresenta um visão mais enxuta e se torna mais facilmente compreensível no que diz respeito aos relacionamentos entre os atores e suas dependências no ambiente organizacional.

Assim, alguns termos básicos devem ser mencionados para adequada compreensão do modelo [21]:

- *Depender*: ator que depende de alguma forma de outro ator;
- *Dependum*: objeto ou elemento de dependência;
- *Dependee*: ator responsável por fornecer a condição necessária para solucionar a dependência.

Desse modo, temos uma relação da forma *Depender* → *Dependum* → *Dependee*.

De acordo com o tipo do *dependum*, o i* dispõe de diferentes tipos de dependências que são detalhados conforme segue [9]:

- Dependência de objetivo: consiste no alcance/satisfação de um *dependum* do tipo objetivo por parte do *dependee* para atender às necessidades do *depender*. Dessa forma, o *dependee* pode se utilizar dos seus próprios meios para alcançar tal objetivo, inclusive pode ocorrer o caso de até não alcançar, o que impacta diretamente o *depender*, por ser tão vulnerável neste aspecto;
- Dependência de recurso: consiste na obtenção de um *dependum* do tipo recurso por parte do *depender* que é disponibilizado pelo *dependee*. Assim, esse recurso é fundamental para que o *depender* possa continuar suas atividades no ambiente

organizacional. A não disponibilidade do recurso pelo *dependee* impacta diretamente nas atividades do *depender*;

- Dependência de tarefa: consiste na execução de um *dependum* do tipo tarefa por parte do *dependee* para que o *depender* possa dar continuidade às suas atividades no ambiente organizacional. No entanto, as diretrizes para executar a tarefa são fornecidas pelo *depender*. Pode ocorrer de a atividade não ser executada pelo *dependee* ou ser executada com atraso por questões de prioridade, o que impacta diretamente nas atividades do *depender*;
- Dependência de objetivo-soft: consiste no alcance/satisfação de um *dependum* do tipo objetivo-soft por parte do *dependee*. Um objetivo-soft é basicamente um objetivo cuja avaliação de satisfação é subjetiva. A não resolução desta dependência também impacta diretamente nas atividades do *depender*.

A figura 2.2 exemplifica um caso amplamente conhecido na literatura de um sistema agendador de reuniões. Nessa figura pode-se ver o relacionamento entre os atores e as dependências entre eles no modelo SD.

O caso mostrado na figura retrata uma modelagem organizacional centralizado no ator do sistema *Agendador de Reunião*. Observe que o foco principal é que o sistema possa atender o objetivo *Reunião Agendada*. Para isso, o *Agendador de Reunião* precisa que o ator *Iniciador de Reunião* (papel representado por algum ator real da organização) execute a tarefa de *Entrar Intervalo de Datas*. Ademais, o sistema precisa que o ator *Participante de Reunião* (papel representado por algum ator real interno ou externo à organização) execute a tarefa de *Entrar Datas Disponíveis*. Com isso, o sistema pode de alguma forma, como realizando cruzamento de datas fornecidas pelos participantes, disponibilizar um recurso que é a *Data Proposta*. Assim, o *Participante de Reunião* pode disponibilizar um recurso *Concordância*, que define se ele está ou não de acordo com a realização da reunião na data proposta.

É importante atentar-se ao fato de que outros objetivos ou objetivos-soft também estão relacionados na modelagem, mas que não possuem relações de dependência com o sistema e sim com outros atores organizacionais.

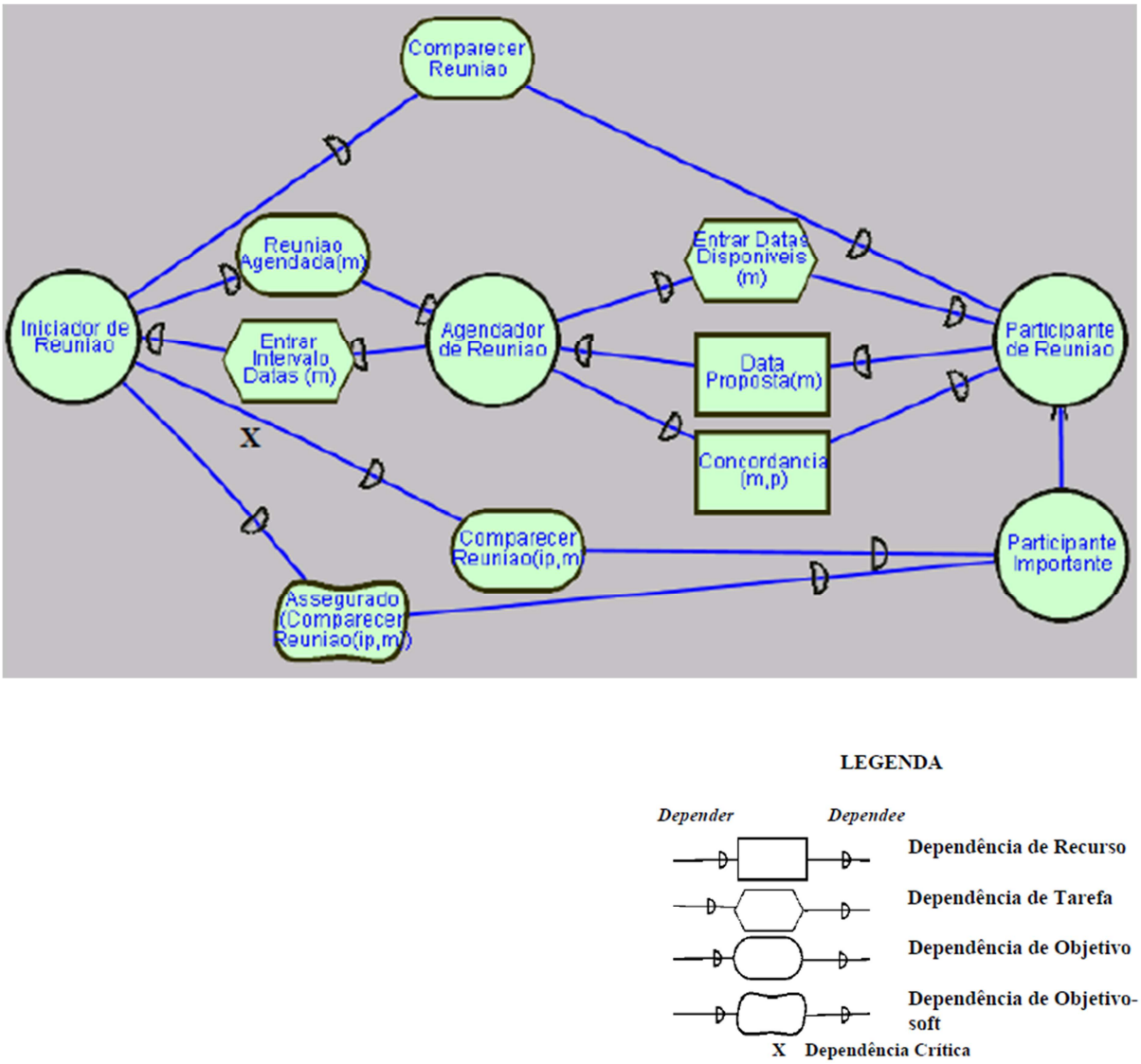

Figura 2.2 Modelo de dependências estratégicas de um agendador de reuniões [4].

Modelo de Razões Estratégicas - SR

Diferentemente do modelo de dependências estratégicas, o modelo de razões estratégicas é mais rico em nível de detalhes, pois estende o modelo SD através da inclusão de elementos que modelam a razão estratégica associada com os atores e suas dependências, razões essas que justificam tais dependências. Uma observação detalhada dessas razões estratégicas é que permite a construção de um SR a partir de um SD [9].

A contribuição do modelo SR na engenharia de requisitos é que este modelo permite que se tenha uma visão adequada da inclusão do sistema e como o mesmo se relaciona com as

atividades/rotinas dos processos organizacionais. Assim, pode-se compreender como o sistema pode criar alternativas ou mesmo dar suporte a essas rotinas. Acrescenta-se neste modelo o conceito de ligações **meio-fim** e ligações de **decomposição de tarefas** que são explicados a seguir [9]:

- Ligações meio-fim: consistem em alcançar algum fim, que pode ser um objetivo, uma tarefa, um objetivo-soft ou um recurso a partir de algum meio, que em geral é uma tarefa;
- Ligações de decomposição de tarefas: consistem em estabelecer uma ligação entre uma tarefa e seus sub-componentes os quais, dependendo do contexto organizacional, podem apresentar dependências com outras decomposições de tarefas de outros atores. Dessa forma, as razões associadas com a tarefa são claramente evidenciadas na sua decomposição.

A figura 2.3 exemplifica o modelo SR como extensão do modelo SD do sistema agendador de reunião. Na figura, é importante observar que o ator *Iniciador de Reunião* planeja executar a tarefa de *Organizar Reunião*. Para que essa tarefa seja executada com êxito, além do comparecimento de participantes à reunião, é necessário que os sub-componentes conectados com ligações de decomposição de tarefa sejam atendidos, ou seja, precisa ser uma tarefa rápida, de pouco esforço (ambos representados como objetivos-soft) e que o objetivo principal de ter a reunião agendada seja alcançado. Basicamente, existem duas opções modeladas através de ligações meio-fim que podem satisfazer esses sub-componentes, executando a tarefa de *Agendar Reunião* manualmente ou *Deixar Agendador Agendar Reuniões*. Nesta segunda opção, vê-se claramente a alternativa em que se encaixa o sistema de software para apoiar o processo organizacional.

O ator *Agendador de Reunião* precisa garantir a execução da atividade *Agendar Reunião Automaticamente* para atender o objetivo de ter a reunião agendada. No entanto, para que essa tarefa seja realizada com sucesso, é necessário que, além de o *Iniciador de Reunião* executar a tarefa *Entrar Intervalo Datas*, as tarefas *Obter Datas Disponíveis* dos participantes da reunião (daí encontra-se a melhor data através da execução da atividade *Cruzar Datas Disponíveis*), *Propor Data* e *Obter Concordância* sejam executadas com êxito.

No lado do ator Participante de Reunião, percebe-se que é fundamental que seja executada a tarefa de *Participar de Reunião*, a qual para ser completada é necessário o participante selecionar datas convenientes com suas atividades, concordar com reunião (levando-se em conta que deve concordar com a data proposta da reunião e o processo de agendamento deve ser de baixo esforço) e então comparecer à reunião.

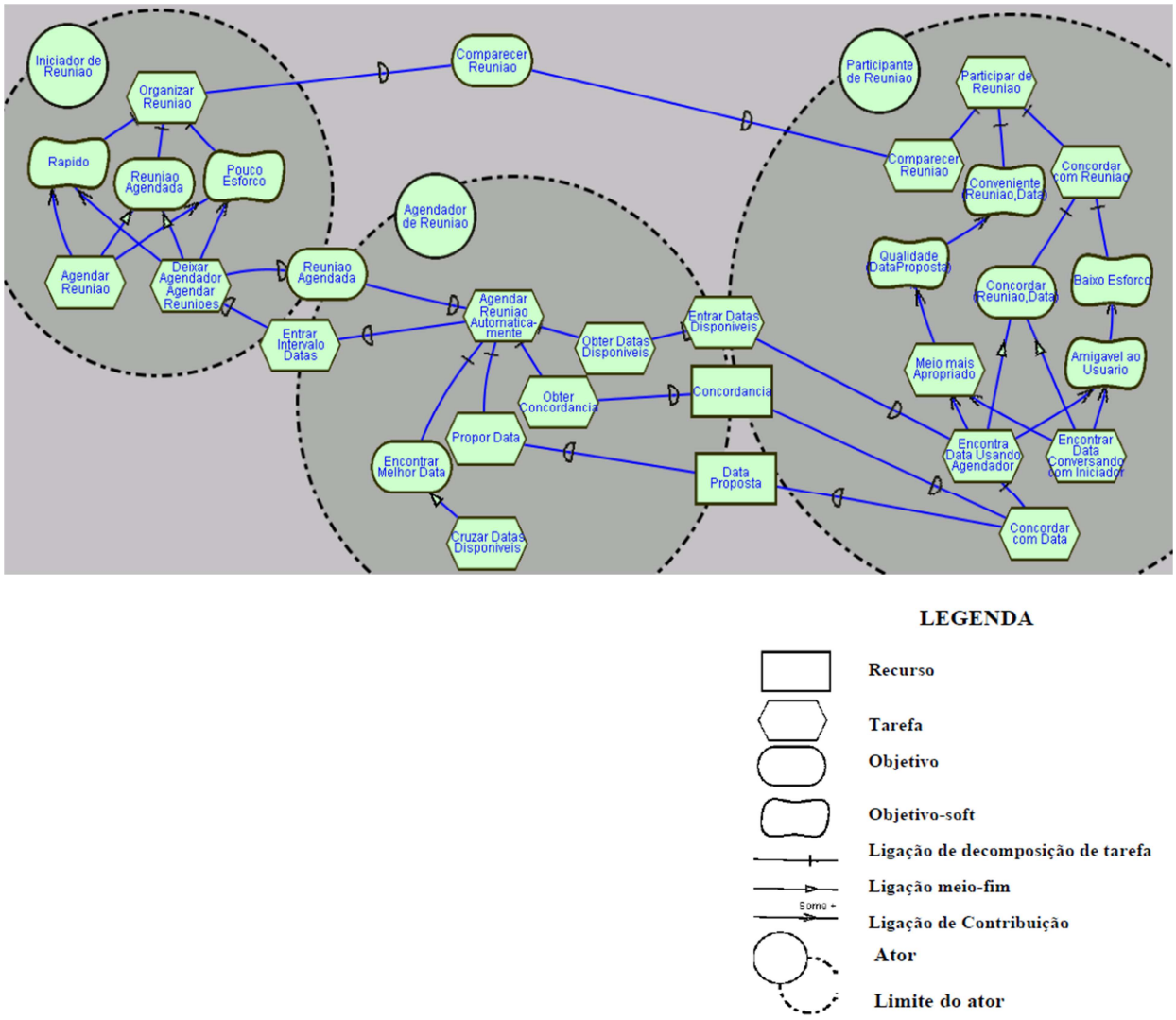

Figura 2.3 Modelo de razões estratégicas de um agendador de reuniões [4].

Na próxima seção, é apresentado o conceito de engenharia de requisitos baseada em processos de negócio. Será detalhado como os processos de negócio de uma organização podem ser modelados para sua utilização na especificação de requisitos.

2.3. Engenharia de Requisitos Baseada em Processos de Negócio

Na literatura, existem diversos conceitos de processos de negócio geralmente bem similares. No entanto, para fins de esclarecimentos, iremos adotar a definição de [16], em que um processo de negócio consiste de um conjunto de atividades que são executadas coordenadamente dentro do ambiente técnico e organizacional. Essas atividades em conjunto destinam-se a atender um determinado objetivo de negócio. Cada processo de negócio é promulgado por uma única organização, mas ele pode interagir com processos de negócio de outras organizações.

Nesse contexto, uma grande dificuldade consiste no processo de gestão dos processos de negócios, isto é, conhecer tanto o ambiente organizacional como a forma com que esse ambiente se relaciona externamente. Esse entendimento é muito importante para que a organização possa ter uma visão mais ampla das situações e, assim, saber priorizar processos de acordo com as estratégias do negócio que vão se alinhar e atingir os objetivos estratégicos da organização [17]. O BPM (*Business Process Management*) é o termo dado a essa gestão de processos de negócio.

Assim como na modelagem baseada em objetivos, a engenharia de requisitos baseada em processos de negócio utiliza a técnica formal de extração de requisitos (ver seção 2.1.2.), uma vez que se apóia no uso de modelos para especificar os requisitos. Entretanto, em ambas abordagens não é uma tarefa fácil desenvolver tais modelos, pois é preciso grande conhecimento do domínio e da complexidade organizacional, ou seja, dos relacionamentos entre os atores/participantes da organização, bem como das tarefas/atividades desempenhadas por eles. Assim, para um mesmo processo organizacional, podem existir modelos diferentes, dependendo da experiência do modelador.

A seguir, será apresentada a modelagem com o BPMN (*Business Process Model and Notation*), notação padrão que apresenta elementos visuais para modelar os processos organizacionais, criada pela BPMI (*Business Process Management Initiative*) em 2004 e que agora é um padrão OMG (*Object Management Group*) [18].

Vale salientar que, dentre as várias notações existentes para modelar processos organizacionais, o BPMN foi escolhido principalmente por sua alta expressividade, facilidade de uso e entendimento, e grande aplicabilidade no mundo dos negócios. Além disso, o BPMN é objeto de estudo do método de [7], o qual será bem detalhado posteriormente no capítulo 4.

2.3.1. Modelagem com o BPMN

A notação BPMN é utilizada para modelar os processos de negócios. Por se tratar de uma notação bastante clara, conhecida e utilizada por todos os usuários do processo de negócio, ganhou destaque como o meio mais adequado para realizar esse tipo de modelagem.

O BPMN visa tornar o duro trabalho de compreender o funcionamento complexo das organizações em uma tarefa mais simples, compreensível pelos *stakeholders* e condizente com a realidade do negócio, dispondo de elementos gráficos que se inter-relacionam modelando todo o processo [7]. Dessa forma, esta notação define um diagrama de modelagem conhecido como BPD (*Business Process Diagram*). Um BPD é constituído basicamente de quatro tipos de objetos gráficos [7]:

- Objetos de fluxo: são os elementos mais importantes na modelagem do processo de negócio. Esses objetos podem ser eventos que podem desencadear em outras atividades ou mesmo ser resultantes das mesmas (eventos de início, intermediário ou de fim), atividades (podem ser tarefas ou sub-processos) e *gateways*[2] (podem ser exclusivos, inclusivos, complexos ou paralelos).
- Objetos de conexão: são elementos que permitem a conexão entre os elementos do BPD. Esses objetos de conexão podem ser fluxos de sequência, os quais podem ser normais ou *default*, fluxos de mensagens ou associações.
- *Swimlanes*: representam os participantes envolvidos no processo de negócio. Podem ser *pools* ou *lanes*.
- Artefatos: são elementos que provêm informações adicionais e podem ser consumidos ou gerados na execução do processo de negócio. Em geral, os artefatos são objetos de dados, grupos e anotações.

Embora o BPMN defina um conjunto de elementos que são utilizados para modelar o processo de negócio, a notação é flexível, pois permite que objetos gráficos sejam definidos de acordo com as necessidades dos usuários do processo de negócio da organização [7].

A figura 2.4 apresenta como são representados os objetos gráficos descritos anteriormente que são definidos na notação BPMN. É importante ressaltar que nem sempre

[2] *Gateways* são elementos de um BPD cuja finalidade é controlar convergência ou divergência do fluxo de sequência.

são usados todos os objetos em uma modelagem. O uso de um ou outro elemento depende da semântica do processo de negócio.

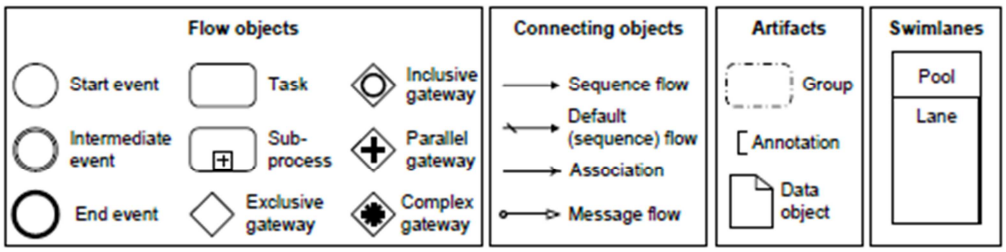

Figura 2.4 Objetos gráficos definidos na notação BPMN [7].

Assim como na seção 2.2.1, o caso do agendador de reuniões será exemplificado, no entanto, usando a notação BPMN. Vale a pena observar o uso dos objetos gráficos já mencionados e notar que, tanto neste modelo como no i*, o objetivo é atingir às metas organizacionais. Neste caso, o objetivo é fazer com que os participantes efetivamente compareçam à reunião (ver figura 2.5). Esse é um diagrama simplificado, pois ele foi obtido do modelo i* a partir das diretrizes de mapeamento apresentadas em [19]. De fato, mais detalhes deveriam ser acrescentados para tornar o processo mais rico e completo.

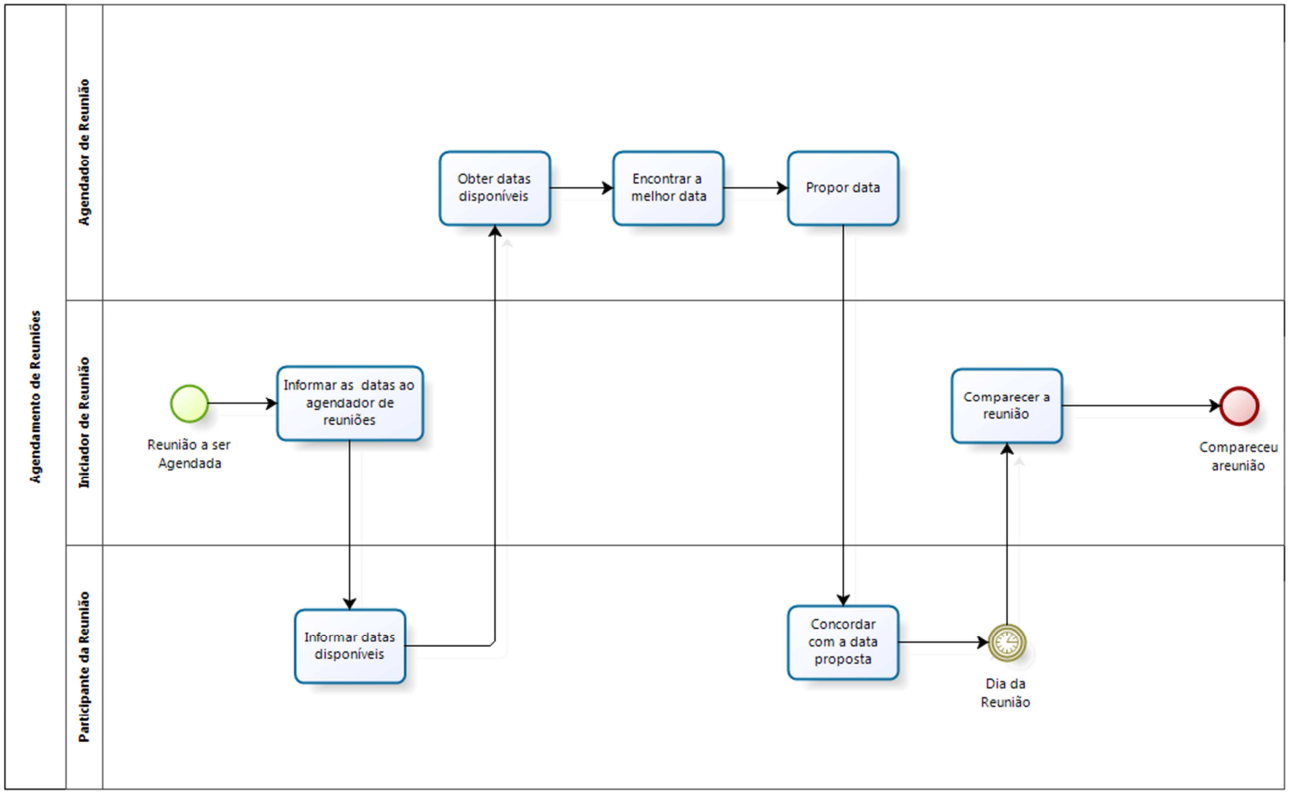

Figura 2.5 Modelagem BPMN de um agendador de reuniões.

Como dito antes, a modelagem de um BPD não é uma tarefa fácil devido aos vários aspectos a serem levados em conta no processo organizacional e muito depende da experiência do modelador com relação àquele nicho organizacional. Para quem "parte do zero", torna-se muito duro projetar este modelo, embora muitos estudiosos, como [7], apresentem *guidelines* que possam auxiliar nessa tarefa. Além disso, muitos esforços têm sido aplicados para obter sistematicamente modelos BPMN do modelo i* e vice-versa, como mencionado anteriormente. No entanto, muitos deles ainda deixam a desejar no processo de mapeamento, com perda de muitas informações úteis ou ainda não foram devidamente validados pra seu uso devido [19]. Isso prejudica a obtenção de modelos i* e BPMN fielmente equivalentes em nível semântico.

Na próxima seção, é apresentado o conceito de casos de uso e cenários. Os casos de uso, os quais são vistos como um conjunto de cenários, permitem compreender melhor o escopo do sistema, facilitar a comunicação entre especialistas e *estakeholders*, e descrever as interações entre usuário e sistema.

2.4. Casos de Uso e Cenários

A compreensão de casos de uso e cenários é fundamental para este trabalho, uma vez que servem de base para a correta interpretação dos conceitos descritos nos capítulos 3 e 4.

No contexto de requisitos de software, há os chamados requisitos funcionais que correspondem às funcionalidades do sistema (para detalhes sobre os tipos de requisitos, ver seção 2.1.1.). Tais requisitos dão origem ao casos de uso, os quais descrevem narrativamente uma seqüência de eventos gerados a partir da ação de um ator (agente externo), ao executar alguma tarefa para alcançar algum objetivo [6]. Os casos de uso, portanto, mapeiam o escopo do sistema, tornam a comunicação com o usuário do sistema mais fluida e contribuem com o gerenciamento do projeto. Nos casos de uso, geralmente não são relatados detalhes específicos de implementação do sistema, uma vez que essas informações fazem parte de etapas que sucedem a engenharia de requisitos, como as fases de projeto e implementação [4].

Um caso de uso pode ser visto como um conjunto de cenários, onde cada cenário contém uma seqüência de passos que representa uma interação entre o usuário e o sistema. É possível imaginar, por exemplo, que uma tarefa seja executada com diferentes caminhos. Assim são os cenários, eles apresentam distintas possibilidades na execução do caso de uso, que levem ao sucesso ou não [8]. Analogamente, os cenários podem ser vistos como sendo

instâncias de um caso de uso, tal como existe o conceito de instância de uma classe no paradigma de programação orientado a objetos [4].

Considerando o contexto atual de execução de um sistema, tem-se que um ou outro cenário de caso de uso pode estar em vigor. Dessa forma, existem dois tipos de cenários: **principal** e **secundário**. No cenário principal, considera-se que estão em execução os passos padrão sem erros ou exceções. É o fluxo de eventos normal do caso de uso. No entanto, como nem "tudo são flores", existem situações que podem ocorrer eventualmente e que fogem do fluxo principal esperado, como é o caso de erros ou exceções. Além disso, pode-se especificar caminhos alternativos para atender o objetivo do caso de uso. Esses casos alternativos e de erros/exceções são descritos como cenários secundários do caso de uso [4].

Para refinar os fluxo de eventos dos casos de uso são introduzidos os conceitos de relacionamentos do tipo <<*include*>>, <<*extend*>> e <<*generalization*>>, que são descritos conforme segue [4]:

- <<*include*>>: consiste em criar um novo caso de uso que pode ser utilizado por outros casos de uso quando se tratar de um comportamento que pode ser comum e repetitivo. Dessa forma, na descrição dos passos do cenário de um caso de uso pode haver uma "chamada" ao caso de uso específico que fora criado mediante este tipo de relacionamento.
- <<*extend*>>: consiste de um caso de uso específico que pode ser "invocado" durante a execução de um outro caso de uso em um cenário secundário, ou seja, em casos opcionais ou condicionais.
- <<*generalization*>>: consiste em atribuir o conceito de herança entre casos de uso. Nesse caso, o filho herda as estruturas e comportamentos do pai, estando livre para modificar, incluir ou eliminar elementos que foram herdados. É similar ao conceito de herança no paradigma de programação orientado a objetos.

Para exemplificar esses conceitos, a figura 2.6 apresenta uma breve ilustração de um diagrama genérico de casos de uso inter-relacionados:

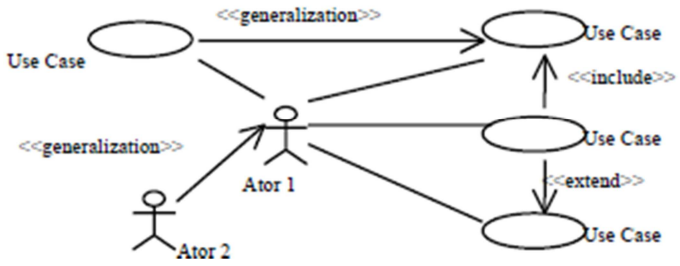

Figura 2.6 Casos de uso inter-relacionados em UML[3] [4].

Apesar de ser uma técnica bastante popular, a geração de casos de uso não é uma tarefa trivial [4]. É necessário descobrir quais são os atores envolvidos, com seus respectivos objetivos, e quais casos de uso trarão algum valor para um determinado ator. Ademais, a descrição dos cenários e a identificação dos relacionamentos do tipo *include* e *extend* exigem um grande conhecimento por parte do engenheiro de requisitos [4].

Uma vez que os casos de uso fundamentam-se no relacionamento dos *stakeholders* com o sistema através da execução de alguma tarefa para atingir algum objetivo, foi definido em [4] diferentes tipos de objetivos associados ao caso de uso: objetivos de usuário (o uso do sistema em suas funcionalidade gera valor agregado ao usuário. Ex: cadastrar um novo cliente numa loja), objetivos de contexto (agrupamento de vários objetivos de usuário em um objetivo mais amplo ou de alto nível. Ex: ampliar a influência da loja no mercado) e objetivos de subfunção (atender esse objetivo é fundamental não para agregar valor ao usuário, mas para que objetivos de usuário sejam alcançados. Ex: logar no sistema).

Em [20], foi definido um *template* de especificação de casos de uso que será utilizado pelo método de Santander [4], a ser detalhado no capítulo 3. Detalhes dos campos presentes nesse *template* podem ser vistos no Apêndice A.

Alguns estudiosos, como é o caso de [7], desenvolveram seus próprios *templates* de especificação de cenários como uma extensão da abordagem de casos de uso, agregando suporte às tarefas do usuário (*Task & Support Descriptions*) [21] e fluxos de informações (*Info Case Approaches*) [22]. A esse tipo especial e estendido de abordagem foi chamado ETD (*Extended Task Description*).

[3] UML (*Unified Modeling Language*) é uma notação que foi inicialmente destinada a modelar sistemas de software orientados a objetos, mas que foi incrementada para dar suporte a outros tipos de modelos, como os diagramas de casos de uso [23].

Detalhando um pouco mais essas extensões agregadas ao conceito de casos de uso, *Task & Support Descriptions* baseia-se na ideia de que os sistemas de software devem suportar as tarefas do usuário. O foco é nos requisitos do domínio, de modo que o sistema possa dar suporte adequado aos usuários na execução das suas tarefas no processo de negócio. Essa abordagem tem uma grande vantagem, por ser fácil de ser compreendida e de ser validada pelos *stakeholders* [21]. Portanto, nesse contexto, existe uma identificação das áreas que são afetadas pelo sistema e das tarefas associadas à cada uma dessas áreas. Cada tarefa, por sua vez, possui um objetivo específico que ou é alcançado ou então a atividade inteira deve ser cancelada.

Info Case Approaches é uma abordagem desenvolvida pela Universidade Federal do Rio de Janeiro que articula modelos de casos de uso com modelos de classes do domínio. O objetivo principal é derivar diagramas de classes a partir de casos de uso. Assim, inclui-se o conceito de fluxo de informação (*information flow*) que são as informações trocadas entre os atores e o sistema em cada caso de uso. Esse fluxo de informação é definido mediante o uso de um gramática formal de dados [22].

Portanto, é claramente perceptível que existe uma diferença básica entre casos de uso e ETDs, pois a primeira tem um foco direto na colaboração entre usuário e sistema, enquanto que a segunda foca na definição das ações do sistema, nas interações com ele e no fluxo de informações trocadas nessas interações [7]. No entanto, pelo fato de ETD ser uma extensão do conceito de caso de uso, com vários elementos comuns, ambas podem ser equiparáveis.

Especificações de cenários utilizando ETDs são utilizadas no método de De la Vara, descrito posteriormente no capítulo 4.

Assim, os capítulos 3 e 4 foram destinados a descrever, respectivamente, o método de Santander para a geração de cenários de casos de uso a partir da modelagem orientada a objetivos i* e o método de De la Vara para a geração de cenários descritos em ETDs, a partir da modelagem de negócio (BPMN).

CAPÍTULO 3

O Método de Santander

"O fracasso não está no insucesso de um objetivo, mas em não ter nenhum objetivo para alcançar"

Benjamin Mays

Este capítulo apresenta o detalhamento de um dos objetos de comparação deste trabalho. Inicialmente, o capítulo aborda uma visão geral sobre a integração entre modelos organizacionais baseados em objetivos com casos de uso. Posteriormente, dedica-se uma seção com a aplicação do método de Santander [4], em que ocorre a integração do *framework* i* com casos de uso. Nessa seção, o método de Santander é destrinchado através de um conjunto de diretrizes cuja exemplificação é realizada no contexto do problema de agendador de reuniões, anteriormente explicado na seção 2.2.1.

3.1. Integração de Modelos de Objetivos e Casos de Uso - Visão Geral

A inserção do conceito de objetivos organizacionais tem sido pauta de muitas pesquisas na área de engenharia de requisitos visando facilitar especificações de requisitos mais completas e alinhadas às metas estratégicas da organização. Percebeu-se com isso que casos de uso podem ser gerados a partir de uma integração com modelos de objetivos [4].

Uma proposta importante que merece ser citada é a do projeto CREWS (*Cooperative Requirements Engineering With Scenarios*), que desenvolveu uma abordagem chamada L'Ecritoire. Nessa abordagem, argumenta-se a favor da utilização dos objetivos organizacionais para elaboração dos cenários de casos de uso. No entanto, por não utilizar heurísticas nem modelos que servem de base para raciocinar sobre os objetivos da organização, o L'Ecritoire é bastante dependente dos conhecimentos do especialista sobre os processos da organização para, num trabalho duro, derivar os casos de uso [24].

Outra proposta que merece atenção é o método conhecido como GBRAM (*Goal-Based Requeriments Analysis Method*) [25]. Este método fundamenta-se na descoberta de objetivos que é uma tarefa essencial para a análise de requisitos. O GBRAM basicamente se divide em duas etapas: análise e refinamento de objetivos. A primeira etapa é composta por

três fases: 1. Explorar as informações disponíveis, 2. Identificar a partir das informações disponíveis os objetivos com os respectivos responsáveis pela satisfação dos mesmos e 3. Organizar os objetivos identificados estabelecendo a relação de dependência entre eles. A segunda etapa também é composta por três fases: 1. Refinar objetivos redundantes/equivalentes, 2. Elaborar uma averiguação dos objetivos para identificar obstáculos e cenários a partir dos quais novos objetivos podem ser encontrados e 3. Operacionalizar os objetivos de modo que sejam mapeados para requisitos operacionais.

A figura 3.1 exemplifica o funcionamento do GBRAM, em que no grupo *Entradas*, existem todas as documentações/informações disponíveis na organização que são utilizados pela fase de exploração da etapa de análise de objetivos. Daí, dá-se prosseguimento às demais fases e etapa conforme explicado anteriormente. Como resultado final da aplicação desse método, tem-se o DRS (Documento de Requisitos de Software) que apresenta os requisitos funcionais e não funcionais do sistema escritos de maneira natural, não ambígua e facilmente compreendia pelos *stakeholders* [4].

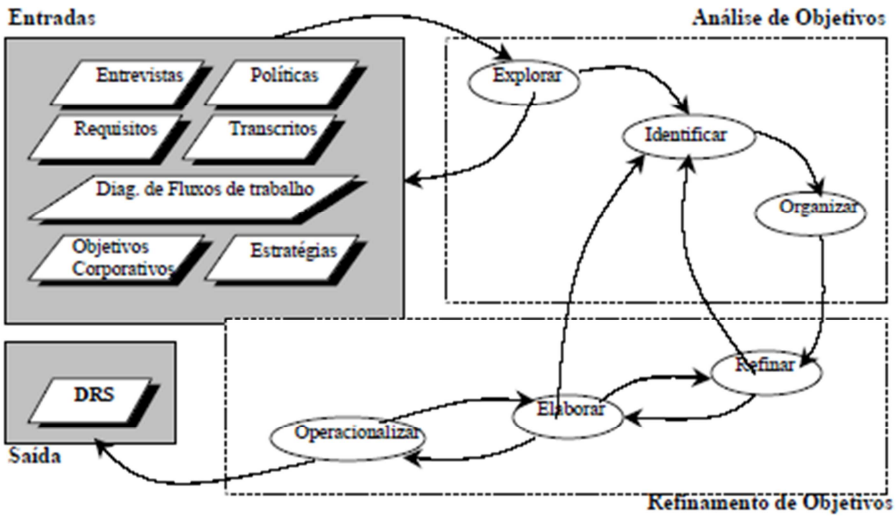

Figura 3.1 Funcionamento do método GBRAM [4].

O Método GBRAM, diferentemente da abordagem L'Ecritoire, define heurísticas para os passos apresentados na figura 3.1, no entanto, essas heurísticas são bastante gerais e não se aplicam aos aspectos específicos de alguns diagramas ou notações como o i*. Nesse contexto, torna-se difícil a aplicação das heurísticas desse método para a obtenção de uma especificação de requisitos através de casos de uso e cenários [4].

A abordagem de Santander, que será detalhada na próxima seção, diferencia-se do projeto CREWS e do método GBRAM positivamente, uma vez que se apóia no uso do modelo de objetivos organizacionais, i*, para derivar os casos de uso do sistema e seus respectivos cenários mediante diretrizes específicas e bem solidificadas. O ponto chave é que, segundo [4], o *framework* i* auxilia a fase inicial da engenharia de requisitos, conhecida como "*early requirements*", através da modelagem dos aspectos organizacionais e da identificação dos "porquês" atrelados aos processos de negócio (razões estratégicas que justificam a necessidade do uso de sistemas de software). A abordagem de Santander fundamenta-se na engenharia de requisitos orientada a objetivos, a qual propõe uma engenharia de requisitos cujo foco principal são os objetivos organizacionais.

3.2. O Método de Santander - Integração do i* com Casos de Uso

Toda organização é composta por relacionamentos entre atores, os quais desempenham papéis importantíssimos nos processos de negócio. Diante da complexidade desses relacionamentos, é fundamental a existência de um modelo capaz de retratar como estão articulados os atores, as tarefas que executam, recursos que disponibilizam, objetivos que pretendem atingir, etc. É aí onde se insere o framework i*, que visa representar toda essa complexidade de maneira mais compreensível por todos os *stakeholders* [9].

O método de Santander que será detalhado nesta seção assume que uma organização, de antemão, dispõe de um modelo i* que englobe todos os seus relacionamentos organizacionais, preferencialmente que apresente o contexto em que se insere o sistema de software no apoio às tarefas diárias da organização. Caso esse modelo seja inexistente, essa deve ser a primeira atividade para que possa ser levado a cabo o processo para a obtenção da especificação dos requisitos numa abordagem orientada a objetivos [4].

O processo de obtenção dos casos de uso acontece a partir de se identificar objetivos de dependências existentes no modelo de dependências estratégicas - SD (ver seção 2.2.1.). Essas dependências podem ser de objetivos, objetivos-soft, tarefas ou recursos. Cada um desses objetivos encontrados dará origem ao objetivo principal do caso de uso que retrata a expectativa do usuário com a execução do caso de uso [4].

O modelo de razões estratégicas - SR (ver seção 2.2.1.), o qual fornece uma visão detalhada do modelo SD contém ricas informações para a obtenção dos fluxos principal e secundários dos casos de uso [4].

O método de Santander ressalta a obtenção de requisitos funcionais do sistema, uma vez que a obtenção de requisitos não funcionais é ainda deficiente, pois o método ainda não trata interdependências entre requisitos não funcionais nem decomposições do mesmo, as quais são muito importantes para identificação dos impactos que poderiam causar nos sistemas de software. No entanto, de forma simplificada, os requisitos não funcionais podem ser obtidos como mapeamento dos objetivos-soft presentes nos modelos do i* [4].

Muitas abordagens como o RUP (*Rational Unified Process*) [26] e Catalysis [27] tratam de casos de uso, no entanto, não levam em consideração as metas da organização para o desenvolvimento de sistemas. Essas abordagens, mesmo dispondo de heurísticas apresentadas na literatura, não deixam os *stakeholders* confortáveis uma vez que não há associação direta com as metas que os mesmo pretendem alcançar. Já há estudos em andamento para utilizar UML para modelar processos de negócio, no entanto, ainda não se tem resultados concretos [4].

A seguir é apresentado o esquema do método de Santander a partir do qual as heurísticas serão aplicadas (ver figura 3.2).

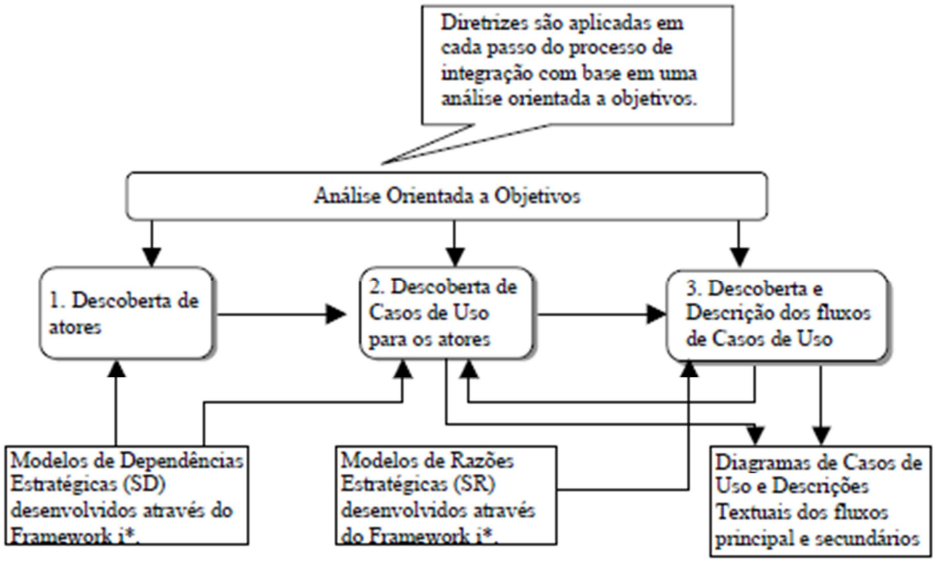

Figura 3.2 Método de Santander para a geração de casos de uso a partir do i* [4].

Na figura anterior, é possível perceber que o método de Santander é aplicado em três passos, todos eles tendo como base uma análise orientada a objetivos em que as diretrizes são aplicadas. Em resumo, esses três passos são detalhados conforme segue:

1. Descoberta de atores: nesse passo, diretrizes são aplicadas para obter os atores dos casos de uso a partir do modelo de dependências estratégicas (SD) do i*;
2. Descoberta de casos de uso para os atores: nesse passo, diretrizes são aplicadas para obter os casos de uso a partir do modelo de dependências estratégicas (SD) do i* para cada um dos atores encontrados no passo anterior;
3. Descoberta e descrição dos fluxo de casos de uso: nesse passo, diretrizes são aplicadas para obter a descrição dos fluxos (primário e secundários) dos casos de uso descobertos no passo anterior a partir do modelo de razões estratégicas (SR) do i*. Com os casos de uso descobertos no passo 2 e com os fluxos descobertos no passo 3, pode-se gerar um diagrama dos casos de uso e uma descrição textual desses fluxos para o preenchimento do *template* proposto por [20].

Vale a pena ressaltar que os passos 2 e 3 podem ser feitos de maneira intercalada e iterativa, uma vez que novos objetivos de casos de uso podem surgir a partir de um refinamento dos cenários gerados.

A seguir, são mostradas as diretrizes (em um total de dez) do método de Santander apoiadas pela técnica GBRAM para obtenção de casos de uso a partir do i*. Essas diretrizes são dispostas em três grupos, sendo cada grupo referente a cada um dos passos da figura 3.2.

3.2.1. Diretrizes do Método de Santander

Nesta seção são mostradas as diretrizes do método de Santander [4], as quais são agrupadas em três grupos: descoberta dos atores, descoberta de casos de uso e descoberta e descrição dos fluxos principal e alternativos dos casos de uso. Para cada grupo, exemplificações do caso do agendador de reuniões serão mostradas a fim de facilitar a compreensão das diretrizes.

Descoberta dos Atores

- Diretriz 1: todo ator em i* deve ser analisado para um possível mapeamento para ator em caso de uso;
- Diretriz 2: inicialmente, deve-se analisar se o ator em i* é externo ao sistema computacional pretendido. Caso o ator seja externo ao sistema, o mesmo é considerado candidato a ator em casos de uso;

- Diretriz 3: deve-se garantir que o ator em i* é candidato a ator em caso de uso, através da seguinte análise, conforme segue:
    - Diretriz 3.1: verificar se existe pelo menos uma dependência do ator analisado em relação ao ator em i* que representa o sistema computacional pretendido;
- Diretriz 4: atores em i*, relacionados através do mecanismo IS-A nos modelos organizacionais e mapeados individualmente para atores em casos de uso (aplicando diretrizes 1, 2 e 3), serão relacionados no diagrama de casos de uso através do relacionamento do tipo <<generalização>>.

Observando a figura 2.2 que apresenta o modelo de dependências estratégicas do caso do agendador de reuniões, pode-se aplicar as diretrizes especificadas (de 1 a 4) para descobrir os atores dos casos de uso relacionados com os atores do modelo em i*.

Do modelo SD, descobre-se que *Iniciador de Reunião*, *Participante de Reunião* e *Participante Importante* são mapeados para atores de casos de uso pois seguem as diretrizes 1, 2 e 3. No caso do *Participante Importante*, além dessas, segue a diretriz 4, sendo mapeado como uma generalização do ator *Participante de Reunião*. É importante ressaltar que o ator *Agendador de Reunião* do modelo SD que representa o sistema, não é mapeado para ator de caso de uso, pois negligencia a diretriz 2.

Descoberta de Casos de Uso

- Diretriz 5: para cada ator descoberto para o sistema pretendido no passo 1, devemos observar todas as suas dependências (*dependum*) do ponto de vista do ator como *dependee***,** em relação ao ator sistema computacional pretendido (sistema computacional → *dependum* → ator)**,** visando descobrir casos de uso para o ator analisado;
    - Diretriz 5.1: deve-se avaliar as dependências do tipo objetivo associadas com o ator;
    - Diretriz 5.2: deve-se avaliar as dependências do tipo tarefa, associadas com o ator;
    - Diretriz 5.3: deve-se avaliar as dependências do tipo recurso, associadas com o ator;

- Diretriz 5.4: deve-se avaliar as dependências do tipo objetivo-soft, associadas com o ator;

- Diretriz 6: analisar a situação especial na qual um ator de sistema (descoberto seguindo as diretrizes do passo 1) possui dependências (como *depender*) em relação ao ator em i* que representa o sistema computacional pretendido ou parte dele. (ator → *dependum* → sistema computacional);
- Diretriz 7: classificar cada caso de uso de acordo com seu tipo de objetivo associado (objetivo contextual, objetivo de usuário, objetivo de subfunção);

Vale ressaltar que, juntamente com a diretriz 6, deve-se verificar se são aplicáveis as diretrizes 5.1, 5.2, 5.3 e 5.4. Tal como no grupo anterior, exemplificam-se as diretrizes deste grupo a partir do modelo de dependências estratégicas do caso do agendador de reuniões. Pode-se aplicar as diretrizes especificadas (de 5 a 7) para descobrir os casos de uso associados com cada ator que foi descoberto.

Do modelo SD, a priori, deve-se observar todos os tipos de dependências existentes (de objetivo, tarefas e recursos) desde que envolvam o sistema, pois essas dependências podem gerar casos de uso do sistema. As dependências do tipo objetivo-soft são mapeadas, posteriormente para requisitos não funcionais do sistema [4]. Como podem existir muitas dependências de atores com o sistema ou mesmo entre os próprios atores, recomenda-se que uma tabela seja criada para organizar as informações. A tabela 3.3 mostra como poderia ser essa tabela, exemplificando os atores *Participante de Reunião* e *Iniciador de Reunião* do caso do agendador de reuniões.

Ator	Dependência	Tipo de dependência	Diretriz a ser Utilizada
Participante de Reunião	Entrar Datas Disponíveis	tarefa	(D 5.2)
Participante de Reunião	Concordância	recurso	(D 5.3)
Participante de Reunião	Data Proposta	recurso	(D 6, D 5.3)
Iniciador de Reunião	Entrar Intervalo Datas	tarefa	(D 5.2)
Iniciador de Reunião	Reunião Agendada	objetivo	(D 6, D 5.1)

Tabela 3.1 Organização das dependências entre atores e sistema [4].

Com essa organização, fica mais fácil alinhar as ideias e identificar os casos de uso associados com cada tipo de dependência conforme a diretriz a ser usada. Uma vez que se tem os casos de uso correspondentes com as dependências (com o mesmo nome inclusive), segue-se a diretriz 7 para classificar os objetivos de cada caso de uso.

Para cada caso de uso, um objetivo está atrelado (em geral com o mesmo nome do caso de uso). A execução do caso de uso deve atender às expectativas do usuário. Portanto, é simples identificar o objetivo, no entanto, nem sempre é fácil classificá-lo, isso vai depender de uma experiência prévia do especialista [4]. Assim, a tabela 3.2 exemplifica a aplicação da diretriz 7 para os objetivos dos casos de uso descobertos:

Tabela 3.2 Aplicação da diretriz 7 para classificar os objetivos dos casos de uso [4].

Ator	Objetivo do Caso de Uso	Classificação do Objetivo
Participante de Reunião	Entrar Datas Disponíveis	Objetivo de Subfunção
Participante de Reunião	Concordância	Objetivo de Subfunção
Participante de Reunião	Data Proposta	Objetivo de Subfunção
Iniciador de Reunião	Entrar Intervalo Datas	Objetivo de Subfunção
Iniciador de Reunião	Reunião Agendada	Objetivo de Usuário

Descoberta e Descrição dos Fluxos Principal e Alternativos dos Casos de Uso

- Diretriz 8: analisar cada ator e seus relacionamentos no modelo de razões estratégicas para extrair informações que possam conduzir à descrição de fluxos principal e alternativos, bem como pré-condições e pós-condições dos casos de uso descobertos para o ator;
    - Diretriz 8.1: analisar os sub-componentes em uma ligação de decomposição de tarefa em um possível mapeamento para passos na descrição do cenário primário (fluxo principal) de casos de uso.
    - Diretriz 8.2: analisar ligações do tipo meio-fim em um possível mapeamento para passos alternativos na descrição de casos de uso.
    - Diretriz 8.3: analisar os relacionamentos de dependências de sub-componentes no modelo de razões estratégicas em relação a outros atores

do sistema. Estas dependências podem originar pré-condições e pós-condições para os casos de uso descobertos.

- Diretriz 9: investigar a possibilidade de derivar novos objetivos de casos de uso a partir da observação dos passos nos cenários (fluxos de eventos) dos casos de uso descobertos. Cada passo de um caso de uso deve ser analisado para verificar a possibilidade do mesmo ser refinado em um novo caso de uso.
    - Diretriz 9.1: inicialmente deve-se averiguar qual é o objetivo que se quer atingir com a ação que o passo representa na descrição do caso de uso;
    - Diretriz 9.2: descoberto o objetivo, deve-se averiguar a necessidade de decompor/refinar o objetivo para que o mesmo possa ser alcançado;
    - Diretriz 9.3: um novo caso de uso será gerado a partir do objetivo analisado, se pudermos definir os passos que representam a necessidade de interações adicionais entre o ator e o sistema para se atingir o sub-objetivo desejado;
    - Diretriz 9.4: adicionalmente, pode-se encontrar novos objetivos e cenários com base na observação dos obstáculos de objetivos já descobertos.
- Diretriz 10: desenvolver o diagrama de casos de uso utilizando os casos de uso descobertos, bem como observando os relacionamentos do tipo <<include>>, <<extend>> e <<generalization>> usados para estruturar as especificações dos casos de uso.

Para exemplificar as diretrizes deste grupo, será utilizado o modelo de razões estratégicas do caso do agendador de reuniões. Pode-se aplicar as diretrizes especificadas (de 8 a 10) para descobrir os fluxos principal e secundários dos casos de uso, bem como pré-condições e pós-condições. A última diretriz, no entanto, referencia a construção do diagrama final de casos de uso.

Do modelo SR, verifica-se, por exemplo, o objetivo *Reunião Agendada*. Tal objetivo foi previamente mapeado como um caso de uso. Para que esse objetivo seja alcançado, o ator *Agendador de Reuniões* que representa o sistema deve realizar com sucesso a atividade *Agendar Reunião Automaticamente*. Esta atividade por sua vez apresenta decomposição de tarefas: *Obter Datas Disponíveis*, *Obter Concordância*, *Propor Data* e *Encontrar Melhor Data*. Portanto, segundo a diretriz 8.1, esses sub-componentes podem ser expressos como fluxo principal para o caso de uso *Reunião Agendada*. Como na decomposição da tarefa

Agendar Reunião Automaticamente não há nenhuma ligação direta do tipo meio-fim, para o caso de uso em questão não haverá a possibilidade de existir fluxos secundários.

Observando os sub-componentes que passam a compor o fluxo principal, tem-se que a tarefa *Obter Datas Disponíveis* depende obrigatoriamente de que o participante da reunião execute a tarefa de *Entrar Datas Disponíveis* (mapeada anteriormente como caso de uso). Da mesma forma, a tarefa *Obter Concordância* depende de que o participante da reunião disponibilize o recurso *Concordância* (mapeado anteriormente como caso de uso). A tarefa *Propor Data* satisfaz a dependência de recurso *Data Proposta* (mapeado anteriormente como caso de uso). Portanto, perceptivelmente existem relacionamentos do tipo <<*include*>> na execução desses passos que compõem o fluxo principal. Outro passo do fluxo principal é resultante do mapeamento do sub-componente *Encontrar Melhor Data*, o qual se satisfaz a partir de um mero cruzamento das datas disponíveis.

A diretriz 9 não é aplicada, pois os passos descritos nos cenários não exigem refinamento. Ou os passos já contêm relacionamentos do tipo <<*include*>> ou então, como é o caso do sub-componente *Encontrar Melhor Data*, não houve necessidade de mapeá-lo em um novo caso de uso, pois seu relacionamento meio-fim apenas determina uma opção que é uma simples forma de encontrar a melhor data através do cruzamento das datas disponíveis.

Por fim, a aplicação da diretriz 10 resulta no diagrama de casos de uso do sistema. As duas figuras a seguir retratam, respectivamente, o diagrama final de casos de uso obtido no contexto do agendador de reuniões e a especificação do caso de uso *Reunião Agendada*.

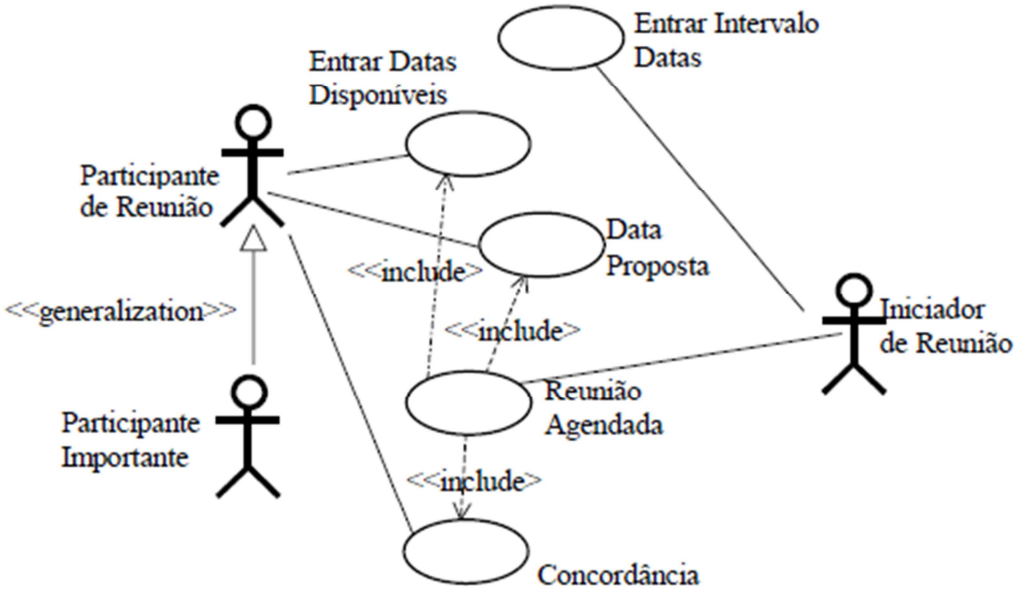

Figura 3.3 Diagrama UML de casos de uso do sistema agendador de reuniões [4].

Caso de Uso: 1 Reunião Agendada

Objetivo no Contexto: Iniciador de Reunião deseja agendar uma reunião com o auxílio do sistema computacional Agendador de Reunião
Escopo: Sistema Agendador de Reunião
Nível: Objetivo de usuário
Precondições: O iniciador deve ter fornecido um intervalo de datas para agendar a reunião (Entrar Intervalo Datas)
Condicao Final de Sucesso: Reunião Agendada
Ator: Iniciador de Reunião

CENÁRIO PRINCIPAL

1. O caso de uso inicia quando com base num intervalo de datas fornecido pelo iniciador de reunião, o sistema solicita dos participantes uma lista de datas possíveis para a reunião (O Caso de Uso ***Entrar Datas Disponíveis*** é incluído <<include>> neste passo).
2. O sistema deve encontrar a melhor data para o agendamento cruzando as informações das datas disponíveis enviadas pelos participantes;
3. O sistema deve propor uma data para agendamento; (O Caso de Uso ***Data Proposta*** é incluído <<include>> neste passo).
4. O agendador de reunião aguarda confirmação (concordância) dos participantes em relação à data proposta encerrando o processo de agendamento quando todos os participantes concordarem com a data. (O Caso de Uso ***Concordância*** é incluído <<include>> neste passo).

EXTENSÕES

INFORMAÇÃO RELACIONADA
Caso de Uso Pai:
Casos de Uso Subordinados:
Entrar Datas Disponíveis
Data Proposta
Concordância

Figura 3.4 Especificação do caso de uso *Reunião Agendada* [4].

Uma observação sobre a ordem dos passos do fluxo principal da figura acima é que ela depende dos conhecimentos do especialista, de modo que a ordem seja exatamente como deve ser, para se obter o objetivo atribuído ao caso de uso.

A seguir, será introduzido o capítulo 4 destinado ao detalhamento do método de De la Vara [7]. Nesse método será mostrado como a modelagem de processos de negócio, através do BPMN, pode auxiliar na especificação de requisitos mediante a geração de cenários estendidos na forma de ETDs.

CAPÍTULO 4

O Método de De la Vara

"Se seus negócios mantêm você tão ocupado que não tem tempo para mais nada, deve ter alguma coisa errada, ou com você ou com seus negócios"

Willian J.H. Boetcker

Este capítulo apresenta o segundo objeto de estudo e de comparação deste trabalho. Inicialmente, o capítulo aborda uma visão geral sobre *As-Is* BPDs e *To-Be* BPDs para contextualizar o meio em que se insere o método de De la Vara [7]. Posteriormente, é mostrada a preparação e modificação de um *To-Be* BPD transformando-o em um *Labelled* BPD e daí, com novas alterações, em um *Enriched* BPD. Por fim, de posse de um *Enriched* BPD, o capítulo apresenta como esta modelagem do processo de negócio pode ser útil para a especificação de requisitos de um sistema de software através de uma ETD. Esse processo de especificação de requisitos é realizado seguindo um conjunto de diretrizes definidas pelo método de De la Vara, as quais são aplicadas no contexto do exemplo agendador de reuniões.

4.1. *As-Is* BPDs e *To-Be* BPDs - Visão Geral

Nesta seção serão apresentados os conceitos de *As-Is* BPDs e *To-Be* BPDs. O entendimento desses conceitos é importante para dar prosseguimento com as mudanças necessárias na modelagem dos processos de negócio para a aplicação do método de Da la Vara.

Um *As-Is* BPD é o termo dado à modelagem, através de um diagrama, da situação atual de um processo de negócio em uma organização. A construção deste diagrama não é tão simples e depende da experiência do modelador sobre o ambiente organizacional. Geralmente, a modelagem é resultante da análise de um conjunto de informações obtidas a partir de artefatos produzidos previamente como entrevistas com funcionários ou mesmo documentações da organização. A correta representação desse diagrama é fundamental para garantir o devido acompanhamento, entendimento e execução das atividades por todos os participantes da organização[7].

A validação dessa modelagem deve ser feita pelos *stakeholders* envolvidos para ratificar que tudo está correto e bem representado, ou seja, que não haja informações faltantes ou não condizentes com o processo de negócio. Caso haja algum problema, novas informações devem ser coletadas e o diagrama deve ser ajustado ou refeito, bem como os artefatos a partir do qual esse diagrama fora criado, para garantir a consistência [7]. A figura 4.1 sumariza o processo para a geração de um *As-Is* BPD.

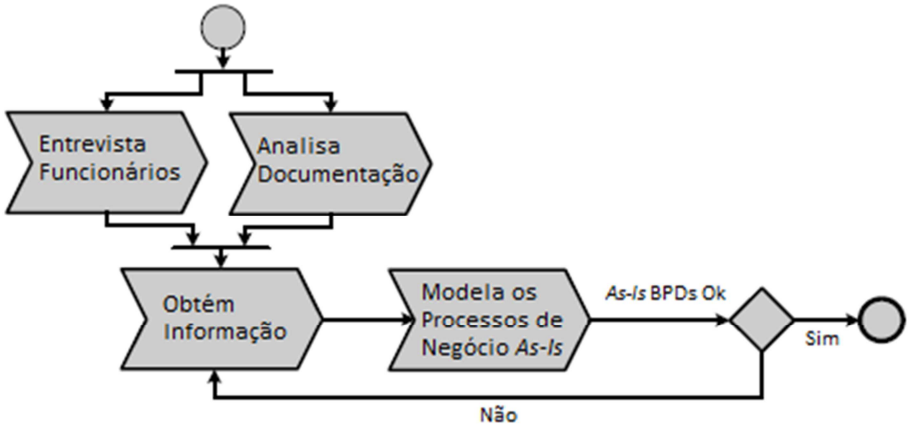

Figura 4.1 Esquematização para a geração de um *As-Is* BPD (adaptado de [7]).

Em [7], existe um conjunto de diretrizes que pode auxiliar na criação de um *As-Is* BPD. Embora essas diretrizes guiem o processo de criação do BPD, uma visão adequada da organização como um todo torna-se essencial para garantir alguns aspectos [7]: (i) corretude da sintaxe e da semântica; (ii) relevância dos objetos representados; (iii) eficiência econômica (custo/benefício e factibilidade do processo modelado); (iv) clareza em relação à leitura e compreensão do modelo; (v) comparabilidade com relação ao andamento dos negócios no passar do tempo, etc.

Uma vez que o BPD correspondente à situação atual da organização tenha sido modelado, pode haver a necessidade da introdução de um novo SI destinado a resolver problemas ou auxiliar na execução das atividades do processo de negócio da organização. Sendo assim, mudanças no diagrama (inclusão e/ou remoção de elementos ou alteração da estrutura do diagrama) podem ocorrer para que o novo sistema proposto possa ser incluído e para que suas devidas responsabilidades e impactos no processo sejam retratados [7]. É nesse contexto que surge o termo *To-Be* BPD.

Assim como o *As-Is* BPD, o *To-Be* BPD também não é tão simples de ser modelado, pois as devidas consequências da introdução do novo SI devem ser bem identificadas. Além

disso, a validação do diagrama pelos *stakeholders* também deve ser realizada. Como o *To-Be* BPD pode envolver a mudança do processo de negócio, é importante que os objetivos do sistema estejam bem claros para que a operacionalização desses objetivos possa ser especificada na modelagem do processo de negócio. Sendo assim, diagramas do tipo objetivos/estratégias (*goals/strategies*), conhecidos como Map [28], podem ser realizados para definir os objetivos do sistema proposto e quais as estratégias para alcançar esses objetivos. Essas estratégias são então operacionalizadas como atividades que serão expressas no BPD [7].

A figura 4.2 exemplifica um diagrama do tipo objetivos/estratégia de uma companhia de desenvolvimento de software que tem vivenciado um problema de atraso na entrega de um software solicitado por um determinado cliente. Este problema ocorre devido à falta de conhecimento sobre o desenvolvimento da versão e, com isso, o objetivo de manter o cliente satisfeito não é atingido. O motivo principal para o atraso é que os desenvolvedores do software acabam sobrecarregados de atividades, de modo que as coisas não andam conforme planejado. Os nós do diagrama representam objetivos a serem atingidos com o novo SI, que é capaz de resolver esse problema da organização. As arestas representam as estratégias, ou seja, as características (*features*) do sistema proposto para resolver o problema [7].

De uma forma geral, o novo SI objetiva auxiliar os desenvolvedores no desenvolvimento do item de trabalho, no fornecimento do status de cada tarefa a ser realizada, no conhecimento sobre o status das versões dessas tarefas e na redução do tempo final de entrega do produto. Assim, evitará atrasos e deixará o cliente mais satisfeito [7].

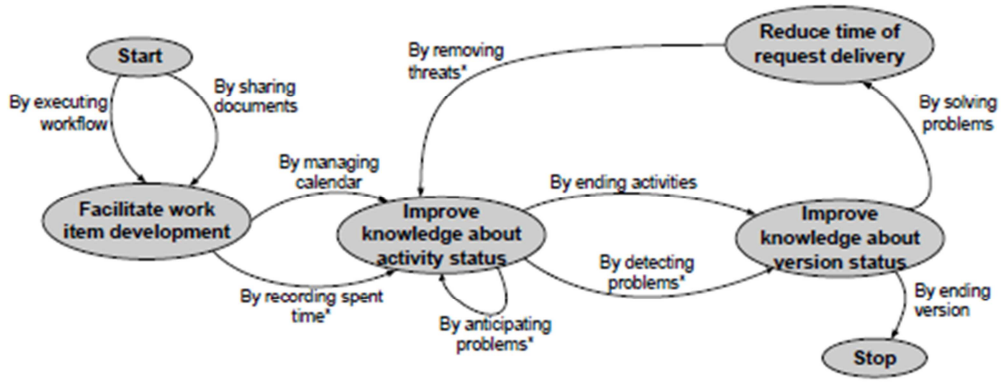

Figura 4.2 Exemplo de um diagrama do tipo objetivos/estratégias [7].

É possível que as estratégias para o alcance dos objetivos precisem ser refinadas. Sendo assim, novos diagramas poderiam ser criados (com mais objetivos e mais estratégias resultantes do refinamento) visando maior detalhamento e esclarecimento sobre as características que o SI deve fornecer e, consequentemente, auxiliando na construção do *To-Be* BPD com a operacionalização dessas estratégias. A figura 4.3 sumariza o processo para a geração de um *To-Be* BPD.

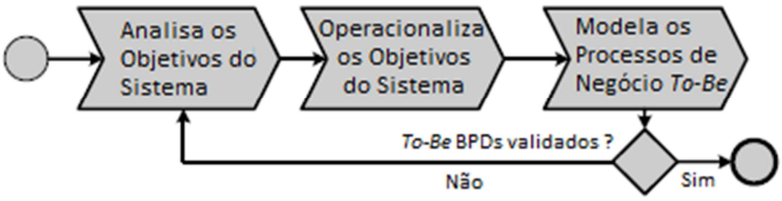

Figura 4.3 Esquematização para a geração de um *To-Be* BPD (adaptado de [7]).

Na próxima seção, serão apresentadas modificações, com a inclusão de novos elementos semânticos ao BPD, propostas por [7]. Essas modificações objetivam preparar um *To-Be* BPD para a posterior aplicação de diretrizes, as quais se destinam a realizar a especificação de requisitos de um sistema de software mediante ETDs. Assim, são introduzidos os conceitos de *Labelled* BPD e *Enriched* BPD.

4.2. *Labelled* BPDs e *Enriched* BPDs

Nesta seção são apresentados dois novos conceitos criados por [7]: *Labelled* BPDs e *Enriched* BPDs. Estes tipos de BPDs serão utilizados para preparar um *To-Be* BPD, de forma que se possa realizar uma extração de informações úteis para a o processo de especificação de requisitos do sistema.

Considerando-se que se dispõe de um *To-Be* BPD, o qual retrata os requisitos operacionais do SI, o conceito de *Labelled* BPD propõe um processo de etiquetagem dos objetos presentes no BPD. Esta etiquetagem permite identificar as corretas atribuições de responsabilidades, isto é, os objetos do fluxo são então classificados conforme os responsáveis por seu controle/execução. Esse processo ajuda os especialistas, a juntamente com os *stakeholders*, especificar as ETDs [7].

Portanto, no BPD, as tarefas, os eventos e os *gateways* são etiquetados de acordo com o suporte que o sistema dá a eles. Quatro tipos de *labels* são atribuídos a esses objetos [7]:

- "**O**": consiste de um objeto que não é controlado/executado pelo sistema (***O**ut of the system*);
- "**L**": consiste de um objeto que é controlado/executado por um sistema legado, já existente (*controlled by a **L**egacy system*);
- "**U**": consiste de um objeto que é controlado/executado por um usuário que interage com o SI (*controlled by a **U**ser*);
- "**IS**": consiste de um objeto que é controlado/executado pelo SI sem intervenção humana (*controlled by the **I**nformation **S**ystem*).

A atribuição desses *labels* nem sempre é trivial. É fácil identificar quando se é fora do sistema ou por um sistema legado. No entanto, nem sempre é possível perceber quando o disparo de um evento deve ser capturado pelo sistema ou pelo usuário, quando a checagem de um gateway deve ser feita pelo sistema ou pelo usuário, ou mesmo quando uma tarefa deve ser executada pelo sistema (com ou sem a iniciativa do usuário) ou só pelo usuário [7]. No caso da execução/controle de uma tarefa, ela pode ser não automatizada (fora do sistema ou controlada pelo sistema legado), parcialmente automatizada (controlada pelo usuário) ou totalmente automatizada (controlada pelo SI).

Ademais, é importante que especialistas e *stakeholders* concordem com as regras de negócio e as entidades do domínio que não foram modeladas no BPD (não foram capturadas pelo modelo, sendo especificadas textualmente), principalmente no que corresponde à parte que será controlada/executada pelo SI.

Uma vez que se tem o BPD modificado com a agregação dos *labels*, vem à tona o conceito de *Enriched* BPD. Basicamente, um BPD enriquecido corresponde à introdução de um novo elemento ao BPD que representa a garantia de que a sequência do fluxo é necessariamente consecutiva (*consecutive flow*). Assim, se o fluxo é consecutivo, ele deve começar e, após sua sequência ordenada, terminar. Se um elemento da sequência é interrompido, todo o fluxo é interrompido. Isso significa que, se dois objetos são conectados por um fluxo consecutivo, eles fazem parte da mesma transação. A representação de um fluxo consecutivo é dada por uma seta com duas pontas. É papel do analista, juntamente com os

stakeholders, identificar os fluxos consecutivos da modelagem, a partir de um certo conhecimento da ordem de execução dos processos de negócio [7].

As figura 4.4 mostra um *Enriched* BPD no contexto do exemplo agendador de reuniões. Nesse BPD, foi feito o processo de atribuição dos *labels* (*Labelled* BPD) e de identificação dos fluxos consecutivos. É importante ressaltar que, apesar de todos os fluxos de sequência da figura terem sido traduzidos como fluxos consecutivos, nem sempre isso ocorre (ex: eventos que não necessariamente acontecem).

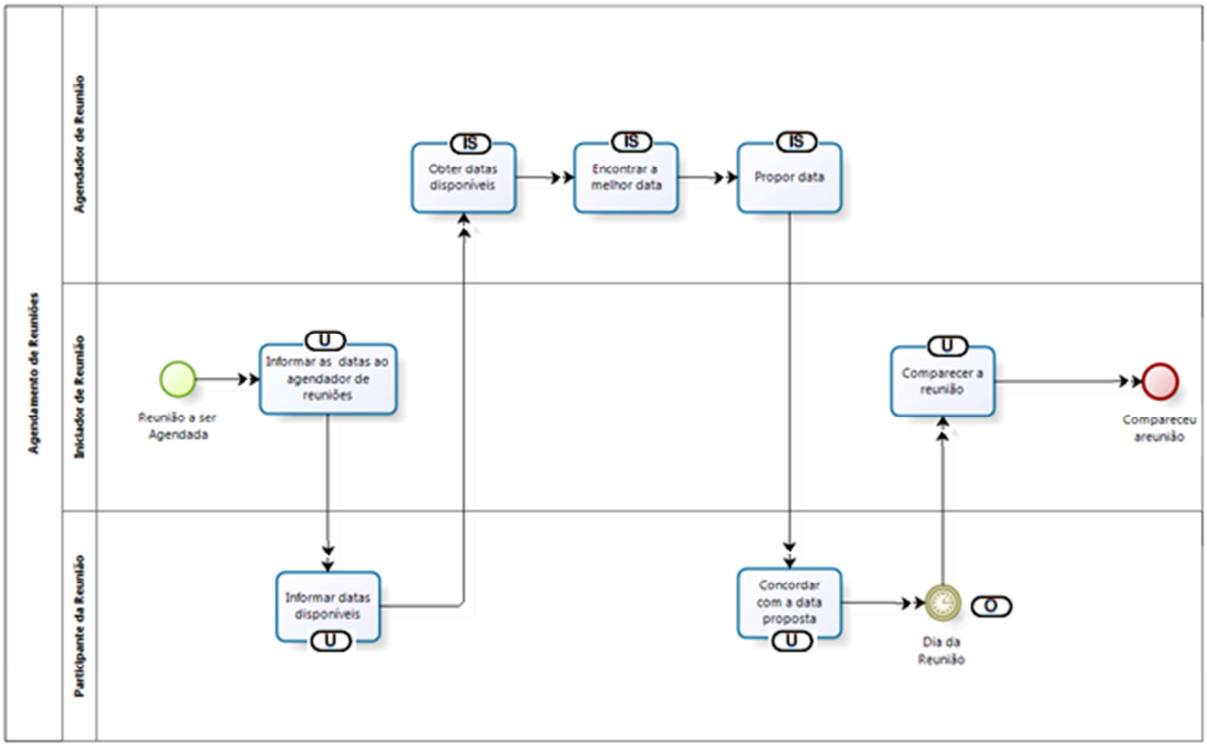

Figura 4.4 *Enriched* BPD do agendador de reuniões após a atribuição dos *labels* e identificação dos fluxos consecutivos.

A próxima seção apresenta as diretrizes do método de De la Vara que são aplicadas ao *Enriched* BPD da figura 4.4 para a obtenção da ETD.

4.3. Diretrizes do Método de De la Vara

Nesta seção, são mostradas as diretrizes do método de De la Vara [7], as quais são organizadas em onze grupos correspondentes às seções que compõem o *template* de uma ETD

(para detalhes sobre essas seções e sobre a estrutura de uma ETD, ver Apêndice B). Esses grupos são: nome; processo de negócio; papel; sub-tarefas; *triggers*; pré-condições; pós-condições; entrada e saída; intenção do usuário e responsabilidade do sistema; fluxos de informação; atributos de qualidade. Complementarmente, para cada grupo de diretrizes, são apresentadas versões parciais do *template* da ETD que são obtidas das mesmas, no contexto do exemplo agendador de reuniões da figura 4.4.

Nome

- Diretriz 1: se uma ETD só suporta uma sub-tarefa, então o nome dela é tal qual o nome da sub-tarefa que é suportada;
- Diretriz 2: o nome de uma ETD deve ser acordada com os *stakeholders* clientes, se ela suporta várias sub-tarefas.

Figura 4.5 Preenchimento do campo que representa o nome da ETD.

Processo de Negócio

- Diretriz 3: o processo de negócio de uma ETD corresponde ao *Enriched* BPD (nome) a partir do qual a ETD foi elicitada. Se o BPD foi modelado para um sub-processo, então o processo de negócio corresponde ao BPD raiz.

Figura 4.6 Preenchimento do campo que representa o processo de negócio da ETD.

Papel

- Diretriz 4: o papel (agente responsável) de uma ETD é o sistema, se todas as sub-tarefas são controladas pelo sistema;
- Diretriz 5: o papel (agente responsável) de uma ETD é o participante no processo de negócio da ETD cuja *lane* contém os objetos de fluxo a partir dos quais as sub-tarefas da ETD que serão controladas pelo usuário são especificadas.

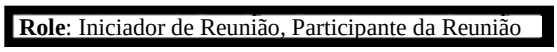

Figura 4.7 Preenchimento do campo que representa o agente responsável da ETD.

Sub-Tarefas

- Diretriz 6: as sub-tarefas de uma ETD são as tarefas BPMN que serão controladas pelo sistema ou por um usuário e os *gateways* e eventos de *"throwing"* que serão controlados pelo usuário.

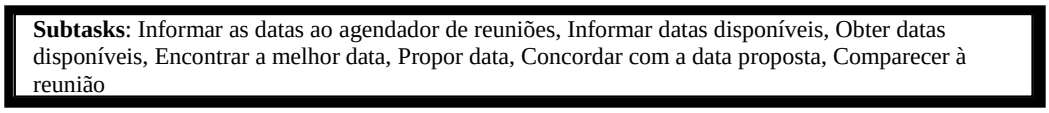

Figura 4.8 Preenchimento do campo que representa as sub-tarefas da ETD.

Triggers

- Diretriz 7: os *triggers* de uma ETD são os eventos de "*catching*" que serão controlados pelo sistema. Se o agente responsável (papel) de uma ETD é o sistema, então os *gateways* que serão controlados pelo sistema também são *triggers*;
- Diretriz 8: os *triggers* de uma ETD podem ser combinados conjuntamente (o cumprimento de todos eles causam a necessidade de execução) e disjuntamente (o cumprimento de algum deles causa a necessidade de execução).

Como o *Enriched* BPD não possui *gateways* nem eventos de "*catching*" controlados pelo sistema, então não existem *triggers* a serem preenchidos no *template* da ETD.

Pré-Condições

- Diretriz 9: as pré-condições de uma ETD são os eventos de "*catching*" que serão controlados por um usuário. Se o agente responsável (papel) de uma ETD não é o sistema, então os *gateways* que serão controlados pelo sistema também são pré-condições;

- Diretriz 10: as pré-condições de uma ETD podem ser combinadas conjuntamente (o cumprimento de todas elas é necessário para que a ETD possa ser executada) e disjuntamente (o cumprimento de alguma delas é necessário para que a ETD possa ser executada).

Como o *Enriched* BPD não possui *gateways* controlados pelo sistema nem eventos de "*catching*" controlados pelo usuário, então não existem pré-condições a serem preenchidas no *template* da ETD.

Pós-Condições

- Diretriz 11: as pós-condições de uma ETD são os eventos de "*throwing*" que serão controlados pelo sistema e os *gateways* que serão controlados pelo sistema e podem fazer as sub-tarefas da ETD iterarem;
- Diretriz 12: as pós-condições de uma ETD podem ser combinadas conjuntamente (todas elas devem ser cumpridas após a ETD ser executada) e disjuntamente (alguma delas deve ser cumprida após a ETD ser executada);
- Diretriz 13: se as pós-condições de uma ETD são combinadas disjuntamente, então deve ser especificado quando elas devem ser cumpridas (isto é, uma condição precede a pós-condição).

Como o *Enriched* BPD não possui *gateways* nem eventos de "*throwing*" controlados pelo sistema, então não existem pós-condições a serem preenchidas no *template* da ETD.

Entrada e Saída

- Diretriz 14: se mais de uma instância de uma entidade de domínio é parte da entrada ou da saída de uma ETD, então as instâncias devem ser diferenciadas por meio de números entre parênteses.

Input		Output	
Domain Entity	**State**	**Domain Entity**	**State**
Data (1) Data (2) Concordância		Data (3)	

Figura 4.9 Preenchimento do campo que representa a entrada e saída da ETD.

Intenção do Usuário e Responsabilidade do Sistema

- Diretriz 15: para cada sub-tarefa de uma ETD que é controlada por um usuário, pelo menos uma ação na intenção do usuário ou na responsabilidade do sistema deve ser especificada;
- Diretriz 16: para cada alternativa e extensão de uma ETD, um nome que identifica a alternativa ou extensão e que se refere à execução sob a qual deve ser executada deve ser especificado;
- Diretriz 17: as ações da interação normal de uma ETD são juntamente ordenadas de acordo com a execução esperada[4] delas;
- Diretriz 18: as ações de uma alternativa são ordenadas por meio de três componentes: o mesmo número da primeira ação da interação normal que elas substituem; uma letra para distinguir dentre as alternativas; e um outro número para ordenar as ações da alternativa;
- Diretriz 19: a ação da interação normal que segue após a última ação de uma alternativa é colocada entre colchetes no final da ação da alternativa;
- Diretriz 20: as ações de uma extensão são ordenadas como as ações de uma alternativa, mas o primeiro número delas corresponde à ação da interação normal que precede as ações da extensão;
- Diretriz 21: em geral, as ações da intenção do usuário correspondem aos seguintes verbos (**selecionar** - informação existente; **introduzir** - nova informação, a qual pode afetar informação existente; **checar** - informação existente);
- Diretriz 22: em geral, as ações de responsabilidade do sistema correspondem aos seguintes verbos (**mostrar** - informação existente; **armazenar** - nova informação; **modificar** - informação existente).

[4] Embora tal ordem represente a sequência esperada dos passos da intenção do usuário e da responsabilidade do sistema, ela pode no fim não representar a sequência atual que leva a cabo uma ETD, ou seja, é só uma possível sequência.

Como o *Enriched* BPD não possui caminhos alternativos (substituem o fluxo normal) nem extensivos (fluxo normal é executado mas podem ocorrer outras ações complementares), então não é possível preencher os campos referentes às alternativas e extensões. No entanto, a figura 4.10 exemplifica o preenchimento dos campos referentes às intenções do usuário e às responsabilidades do sistema.

User Intention	System Responsibility
Normal Interaction	
1. Iniciador de reunião introduz intervalo de datas no sistema	
	2. Sistema armazena intervalo de datas
3. Participante da reunião introduz as suas datas disponíveis para a reunião	
	4. Sistema armazena as datas disponíveis
	5. Sistema encontra a melhor data
	6. Sistema mostra a proposta de data ao participante da reunião
7. Participante da reunião envia sua concordância	
	8. Sistema recebe e armazena concordância
9. Participante comparece à reunião	

Figura 4.10 Preenchimento do campo que representa a interação normal entre usuário e sistema.

Fluxos de Informação

- Diretriz 23: um fluxo de entrada deve ser especificado para cada ação da intenção do usuário no qual o usuário tem que selecionar informações vindas do sistema ou introduzir novas informações no sistema;

- Diretriz 24: um fluxo de saída tem que ser especificado para cada ação de responsabilidade do sistema no qual o sistema tem que mostrar informações aos usuários;
- Diretriz 25: um fluxo autônomo tem que ser especificado para cada ação da intenção do usuário ou de responsabilidade do sistema no qual o sistema de que armazenar ou modificar alguma informação sem entrada dos usuários;
- Diretriz 26: para cada fluxo de informação, a expressão de dados de entrada ou de saída é especificada com base na gramática BNF para especificação de fluxo de informação;
- Diretriz 27: para cada fluxo de informação de uma ETD, as entidades de domínio que são usadas devem fazer parte da entrada ou saída da ETD;
- Diretriz 28: as entidade de domínio e os atributos de um fluxo autônomo de uma ETD devem fazer parte de um fluxo de saída de alguma ETD (a mesma ETD ou uma outra).

Como não existem fluxos autônomos, já que o sistema não armazena nem modifica informações sem a entrada dos usuários, não é possível visualizar a aplicação das diretrizes 25 e 28. No entanto, a aplicação das demais diretrizes deste grupo podem ser percebidas na figura 4.11. Para detalhes sobre a gramática BNF, para especificação de fluxo de informação, ver Apêndice B.

User Intention	System Responsibility	Information Flows
Normal Interaction		
1. Iniciador de reunião introduz intervalo de datas no sistema		→ 1{Data(1)/dia+mes+ano/}n
	2. Sistema armazena intervalo de datas	
3. Participante da reunião introduz as suas datas disponíveis para a reunião		→ 1{Data(2)/dia+mes+ano/}n
	4. Sistema armazena as datas disponíveis	
	5. Sistema encontra a melhor data	
	6. Sistema mostra a proposta de data ao participante da reunião	← Data(3)
7. Participante da reunião envia sua concordância		→ Concordância/valor/
	8. Sistema recebe e armazena concordância	
9. Participante comparece à reunião		

Figura 4.11 Preenchimento do campo que representa o fluxos de informação (*information flows*) presentes na interação normal entre usuário e sistema.

Atributos de Qualidade

- Diretriz 29: para cada atributo de qualidade de uma ETD, um tipo de característica e sub-característica do padrão ISO 9126-1 deve ser especificado;
- Diretriz 30: se um sistema externo participa em uma ETD como um usuário, então o atributo de qualidade de "Funcionalidade/Interoperabilidade" deve ser especificado e deve se referir ao sistema;
- Diretriz 31: se ambos, um usuário humano (que irá corresponder ao agente responsável - papel - da ETD) e um sistema externo participam em uma ETD, então os passos da interação executados pelo sistema devem ser indicados por meio de um atributo de qualidade de "Funcionalidade/Interoperabilidade".

Como não há sistemas externos, que atuam como um usuário, participando da ETD, não é possível preencher os atributos de qualidade no *template* da ETD.

Embora as diretrizes apresentadas possibilitem o preenchimento dos campos de uma ETD, alguns campos não são cobertos por essas diretrizes (ex.: *Frequency*, *Critical*, *Business Rules* e a coluna *State*, dos campos *Input* e *Output*), ficando a critério do especialista identificar a informação adequada, baseando-se na sua experiência ou em documentações da organização.

É importante ressaltar que uma ETD deve ser validada pelos *stakeholders*, para garantir que tudo está conforme o que se espera do sistema. Caso não esteja, deve-se verificar o que deve ser alterado desde o *Labelled* BPD.

Em resumo, a figura 4.12 e 4.13, mostram, respectivamente, os passos discutidos neste capítulo para a obtenção da especificação de ETDs e a ETD completa para o exemplo agendador de reuniões.

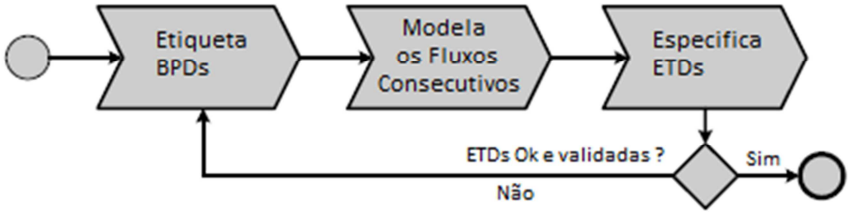

Figura 4.12 Esquematização para o processo de especificação de ETDs (adaptado de [7]).

Extended Task Description: Agendamento de Reuniões			
Business Process: Agendamento de Reuniões		**Role**: Iniciador de Reunião,Participante da Reunião	
Subtasks: Informar as datas ao agendador de reuniões, Informar datas disponíveis, Obter datas disponíveis, Encontrar a melhor data, Propor data, Concordar com a data proposta, Comparecer à reunião			
Triggers: - **Preconditions**: - **Postconditions**: -			
Frequency: 1 vez por mês **Critical**: -			
Input		**Output**	
Domain Entity	**State**	**Domain Entity**	**State**
Data(1) Data(2) Concordância	- - Ready	Data(3)	-
Business Rules • A reunião só é agendada se o valor da concordância de todos os participantes for positiva, i.e, todos desejam participar da reunião. • A melhor data para a reunião deve ser encontrada através de um cruzamento das datas do iniciador e dos participantes.			
User Intention	**System Responsibility**	**Information Flows**	
Normal Interaction			
1. Iniciador de reunião introduz intervalo de datas no sistema	2. Sistema armazena o intervalo de datas	→ 1{Data(1)/dia+mes+ano/}n	
3. Participante da reunião introduz as suas datas disponíveis para a reunião	4. Sistema armazena as datas disponíveis	→ 1{Data(2)/dia+mes+ano/}n	
	5. Sistema encontra a melhor data		
	6. Sistema mostra a proposta de data ao participante da reunião	← Data(3)	
7. Participante da reunião envia sua concordância		→ Concordância/valor/	
9. Participante comparece à reunião	8. Sistema recebe e armazena concordância		
Alternatives			
Extensions			
Quality Attributes			

Figura 4.13 ETD completa para o exemplo agendador de reuniões.

A seguir, inicia-se o capítulo 5, que se fundamenta em uma análise comparativa dos métodos até aqui apresentados.

CAPÍTULO 5

Comparação dos Métodos

"Nada é bom ou mau se não for por comparação"

Thomas Fuller

Este capítulo apresenta uma análise comparativa dos métodos mostrados nos dois capítulos anteriores. Embora bem diferentes, o método de Santander e o método de De la Vara visam apoiar a engenharia de requisitos, com a geração de casos de uso (normais ou estendidos através de ETDs), de modo a favorecer uma especificação de requisitos mais correta, completa e alinhada às metas da organização. Inicialmente, este capítulo apresenta uma visão geral sobre a necessidade e importância da comparação dos métodos. Posteriormente, dedica-se uma seção à definição das métricas de avaliação e, por fim, o capítulo se encerra com a avaliação dos dois métodos.

5.1. Os Métodos de Santander e De la Vara - Por que Compará-los ?

Um processo de comparação é muito importante quando se deseja escolher a melhor entre uma ou outra opção disponível, independentemente do objeto em questão. Por exemplo, um caso simples de comparação acontece quando alguém precisa comprar uma roupa para ir a um baile de formatura. Geralmente, a escolha de um "bom" terno ou vestido pode considerar vários aspectos, como, por exemplo, analisando o preço da peça, o tamanho, a beleza, a adequação ao evento, dentre outros. Até nos aspectos mais simples da vida, um processo de comparação é feito e, mesmo que automaticamente ou informalmente, métricas de comparação são definidas para avaliar qual objeto é melhor que outro para determinado contexto, podendo ser esse objeto qualquer coisa material ou imaterial.

Na engenharia de requisitos não é diferente. A cada dia, muitas pesquisas têm sido realizadas para desenvolver métodos capazes de contribuir com o processo de especificação de requisitos de software. Pretende-se que esses requisitos sejam os mais corretos e completos possíveis e alinhados às metas da organização. Desse modo, o SI pode resolver problemas enfrentados no dia a dia da organização ou mesmo apoiar na execução de suas atividades.

Nesse contexto, enquadram-se dois importantes métodos conhecidos na literatura, os quais se destinam a especificar cenários de casos de uso, a partir da modelagem de objetivos organizacionais ou da modelagem de processos de negócio. No primeiro caso, está o método de Santander [4], que lança mão do *framework* i* para a geração de casos de uso. No segundo caso, está o método de De la Vara, que se utiliza da modelagem dos processos de negócio da organização, através de uma notação estendida do BPMN, para a geração de casos de uso estendidos através de ETDs [7].

Sendo assim, um estudo comparativo desses dois métodos ajuda a indicar o melhor método, de acordo com algumas métricas de avaliação, para a especificação de casos de uso e geração de cenários. Isto possibilita aos engenheiros de requisitos, bem como aos demais envolvidos, a escolha de um método que permita obter a especificação de requisitos de software mais correta, completa e alinhada às metas da organização.

A próxima seção formaliza a definição das métricas utilizadas para a comparação dos dois métodos estudados neste trabalho.

5.2. Definição das Métricas de Avaliação dos Métodos

Neste trabalho, a avaliação dos métodos de Santander e de De la Vara se dá a partir de métricas definidas utilizando o conceito GQM [5](*Goal Question Metric*) [10].

Inicialmente, é preciso identificar os objetivos da avaliação. Sendo assim, foram definidos quatro objetivos principais:

- Objetivo 1: avaliar a corretude dos casos de uso gerados;
- Objetivo 2: avaliar a completude dos casos de uso gerados;
- Objetivo 3: avaliar a complexidade do método;
- Objetivo 4: avaliar o alinhamento dos requisitos especificados com as metas organizacionais.

Uma vez que os objetivos foram identificados, dá-se prosseguimento à elaboração de questões, que servirão de base para a definição das métricas. Portanto, a tabela 5.1 mostra, para cada objetivo, as questões correspondentes e as métricas associadas a elas. Essas

[5] GQM (*Goal Question Metric*) é uma abordagem para elaborar objetivos e definir questões que serão associadas às métricas [10].

métricas são os elementos fundamentais para a avaliação do método de geração de casos de uso, segundo os objetivos definidos.

Além das questões com o nome das suas respectivas métricas, a tabela inclui, para cada métrica, os seguintes campos:

- Definição rápida: texto informal que resume a definição da métrica;
- Cálculo da métrica: texto que apresenta como deve ser feito o cálculo da métrica. Nos casos em que a métrica retrata uma possibilidade ou a existência de algo, nenhuma forma de cálculo é aplicada. Neste caso, a métrica deve ser utilizada verificando tal possibilidade/existência.
- Pré-condições: texto que apresenta as pré-condições para que o cálculo da métrica seja realizado;
- Comentários: texto que apresenta explicações adicionais sobre a métrica ou recomendações que possam ajudar na interpretação das métricas para uma correta avaliação dos objetivos.

Para uma completa avaliação dos objetivos propostos, é importante ressaltar a necessidade de uma avaliação qualitativa, mediante a elaboração e aplicação de um questionário que seria respondido pelos *stakeholders* (ver Apêndice C). No entanto, devido ao tempo curto disponível para a elaboração deste trabalho, não foi possível realizar esta avaliação qualitativa, de modo que a avaliação foi feita apenas quantitativamente.

Tabela 5.1 Definição das métricas.

Objetivo 1: corretude dos casos de uso gerados	
Q1: O caso de uso deixa explícito qual o valor que sua execução trará ao usuário?	
Nome da métrica	**EOSCU** - Existência, no *template*, do objetivo que é satisfeito com a execução do caso de uso.
Definição rápida	Existência, no *template*, de campo associado ao objetivo que é satisfeito com a execução do caso de uso. Esta métrica visa avaliar se existe uma preocupação em especificar casos de uso que alcancem os objetivos organizacionais.
Cálculo da métrica	Não se aplica.

Pré-condições	Não há pré-condições.
Comentários	Se o *template* tiver o campo objetivo, a resposta é sim. Se houver algum campo, a partir do qual se depreende, facilmente, o objetivo, a resposta também é sim. Caso contrário, a resposta é não.
Q2: Quão validado foi o processo de obtenção dos casos de uso, bem como o template final resultante desse processo ?	
Nome da métrica	**QVN** - Quantidade de validações necessárias, desde o modelo-base até o *template* final de especificações de casos de uso.
Definição rápida	Quantidade de validações necessárias, realizadas pelos *stakeholders* nos artefatos gerados, desde a modelagem organizacional/processos de negócio, até o *template* final resultante de especificações de casos de uso. Esta métrica visa avaliar se existe uma preocupação em validar esses artefatos, a fim de garantir a corretude do resultado final.
Cálculo da métrica	Contabiliza-se a quantidade de validações necessárias, realizadas pelos *stakeholders*.
Pré-condições	Não há pré-condições.
Comentários	É importante uma constante validação realizada pelos *stakeholders*, para garantir a conformidade/adequação dos artefatos gerados, contribuindo, portanto, com a corretude da especificação de casos de uso gerados.
Objetivo 2: completude dos casos de uso gerados	
Q3: Quão detalhados são os modelos-base, a partir dos quais se obtêm os casos de uso ?	
Nome da métrica	**QEMB** - Quantidade de elementos que o modelo-base (modelo organizacional/de processos de negócio) dispõe.
Definição rápida	Quantidade de elementos que o modelo-base, a partir do qual se obtêm os casos de uso, dispõe na modelagem organizacional/processos de negócio.
Cálculo da métrica	Contabiliza-se a quantidade de elementos que o modelo-base dispõe.
Pré-condições	Não há pré-condições.
Comentários	É importante o uso de um modelo-base que disponha de vários elementos, a fim de que mais detalhes sejam capturados para a geração dos casos de uso.
Q4: Quão abrangentes são as diretrizes do método para guiar o processo de obtenção dos casos de uso ?	
Nome da métrica	**PCTPD** - Percentual de campos do *template* preenchidos com as diretrizes.

Definição rápida	Percentual de campos do *template* de especificação de caso de uso preenchidos com as diretrizes definidas pelo método.
Cálculo da métrica	Calcula-se o percentual da seguinte forma: QCPD/QC. Onde: QCPD corresponde à quantidade de campos preenchidos com as diretrizes, e QC é a quantidade total de campos, obrigatórios ou opcionais, que o *template* dispõe.
Pré-condições	É necessário que o *template* disponha de pelo menos um campo.
Comentários	Um *template* que disponha de muitos campos preenchidos com o uso das diretrizes, permite maior rapidez e sistematização no preenchimento.
Q5: Quão completos são os cenários dos casos de uso gerados com as diretrizes ?	
Nome da métrica	**QPC** - Quantidade de passos dos cenários gerados.
Definição rápida	Quantidade de passos dos cenários, principal e secundários, gerados após a aplicação das diretrizes do método.
Cálculo da métrica	Calcula-se a quantidade de passos dos cenários gerados da seguinte forma: QPCP + QPCS. Onde: QPCP corresponde à quantidade de passos no cenário principal, e QPCS é a quantidade total de passos no(s) cenário(s) secundário(s).
Pré-condições	Não há pré-condições.
Comentários	Quanto mais passos tiverem os cenários, mais completos serão.
Q6: Os casos de uso gerados dão suporte ao reuso de passos de cenários ?	
Nome da métrica	**PUI** - Possibilidade de uso de *include* nas ações normais dos cenários.
Definição rápida	O uso de *include* nas ações normais (passos) dos cenários é suportado pelos casos de uso gerados.
Cálculo da métrica	Não se aplica.
Pré-condições	Não há pré-condições.
Comentários	O uso de *include* torna a especificação dos cenários mais sucinta (reuso de passos de cenários).
Q7: Os templates de casos de uso dão suporte à descrição de ações alternativas ou extensivas ao cenário principal ?	
Nome da métrica	**ECTAAE** - Existência de campo, no *template*, para as ações alternativas ou extensivas.

Definição rápida	Existência de campo, no *template* de especificação de casos de uso, que permita descrever as ações alternativas ou extensivas ao cenário principal.
Cálculo da métrica	Não se aplica.
Pré-condições	Não há pré-condições.
Comentários	Os cenários secundários, quando presentes, tornam a especificação do caso de uso mais completa.
Q8: Quão completos são os templates de casos de uso gerados ?	
Nome da métrica	**QCPPT** - Quantidade de campos para preenchimento no *template*.
Definição rápida	Quantidade total de campos possíveis para preenchimento (obrigatórios ou opcionais), disponibilizados no *template* de especificação de caso de uso.
Cálculo da métrica	Contabiliza-se a quantidade de campos que o *template* dispõe.
Pré-condições	Não há pré-condições.
Comentários	Quanto mais campos o *template* possuir, mais rica será a especificação de caso de uso.
Q9: Quanto suporte à diagramatização dos casos de uso gerados o método oferece ?	
Nome da métrica	**QDGDCU** - Quantidade de diretrizes que auxiliam o processo de geração do diagrama de casos de uso.
Definição rápida	Quantidade de diretrizes que o método oferece para a obtenção de um diagrama de casos de uso.
Cálculo da métrica	Contabiliza-se a quantidade de diretrizes que auxiliam na obtenção do diagrama de casos de uso.
Pré-condições	Não há pré-condições.
Comentários	Um diagrama de casos de uso é importante para auxiliar a comunicação entre analistas e *stakeholders*.
Objetivo 3: complexidade do método	
Q10: Quão complexo é o método em quantidade de diretrizes a serem seguidas ?	
Nome da métrica	**QDSA** - Quantidade de diretrizes e sub-diretrizes a serem atentadas (executadas).
Definição rápida	Quantidade de diretrizes e sub-diretrizes que o método dispõe para guiar

	o processo de especificação dos casos de uso.
Cálculo da métrica	Contabiliza-se a quantidade de diretrizes e sub-diretrizes do método.
Pré-condições	Não há pré-condições.
Comentários	Quanto mais diretrizes e sub-diretrizes houver, mais complexo será o processo de especificação dos casos de uso. Isto, porque vai-se gerar uma demanda maior de tempo e de atenção.
Q11: Quão sistemático é o método ?	
Nome da métrica	Mesmo de **Q4**.
Definição rápida	Mesma de **Q4**.
Cálculo da métrica	Mesmo de **Q4**.
Pré-condições	Mesmas de **Q4**.
Comentários	A métrica definida em **Q4** é aplicada tanto na avaliação de completude, quanto na avaliação de complexidade. No primeiro caso, avalia-se a completude do método em função da cobertura das suas diretrizes em relação aos campos a serem preenchidos no *template*. No segundo caso, a presença de campos "livres", cujo preenchimento não depende de diretrizes, é menos sistemático, exige maiores conhecimentos por parte do analista e obtenção de informações de outros documentos/artefatos da organização.
Objetivo 4: alinhamento dos requisitos especificados com as metas organizacionais	
Q12: Os casos de uso gerados estão de acordo com as metas organizacionais ?	
Nome da métrica	Mesmo de **Q1**.
Definição rápida	Mesma de **Q1**.
Cálculo da métrica	Mesmo de **Q1**.
Pré-condições	Mesmas de **Q1**.
Comentários	Mesmos de **Q1**. Ademais, é importante mencionar que a métrica definida em **Q1** é aplicada tanto na avaliação de corretude, quanto na avaliação de alinhamento dos requisitos especificados com as metas organizacionais. No primeiro caso, o campo associado ao objetivo garante que o caso de uso está correto (conforme desejado pelo cliente). No segundo caso, a presença do objetivo auxilia a identificar qual meta organizacional está sendo satisfeita com o caso de uso.

Q13: Quão justificável é a presença dos casos de uso do sistema em função das metas organizacionais ?	
Nome da métrica	**PCUGRO** - Percentual de casos de uso gerados relacionados a objetivos.
Definição rápida	Percentual de casos de uso que possuem uma relação direta com (ou satisfaz) um objetivo da organização.
Cálculo da métrica	Calcula-se o percentual da seguinte forma: QCURDSO/QCU. Onde: QCURDSO corresponde à quantidade de casos de uso que possuem uma relação direta com (ou satisfaz) um objetivo da organização, e QCU é a quantidade total de casos de uso gerados.
Pré-condições	É necessário que, pelo menos, um caso de uso seja gerado com a aplicação das diretrizes do método.
Comentários	Quanto mais casos de uso associados a objetivos houver, mais alinhados eles estarão às metas organizacionais.
Q14: O processo de obtenção dos casos de uso foi validado segundo às expectativas dos stakeholders para alcançar as metas organizacionais ?	
Nome da métrica	Mesmo de **Q2**.
Definição rápida	Mesma de **Q2**.
Cálculo da métrica	Mesmo de **Q2**.
Pré-condições	Mesmas de **Q2**.
Comentários	Mesmos de **Q2**. Ademais, é importante mencionar que a métrica definida em **Q2** é aplicada tanto na avaliação de corretude, quanto na avaliação de alinhamento dos requisitos especificados com as metas organizacionais. No primeiro caso, saber a quantidade de validações necessárias, realizadas pelos *stakeholders* nos artefatos gerados, contribui na conformidade/adequação dos artefatos. No segundo caso, contribui na ratificação dos artefatos, com relação às expectativas dos *stakeholders* para alcançar as metas da organização.

Uma vez que os objetivos e métricas foram definidos, a próxima seção apresenta os resultados da avaliação dos métodos de Santander e de De la Vara, no contexto do exemplo agendador de reuniões.

5.3. Avaliação dos Métodos

Os métodos estudados neste trabalho serão avaliados nesta seção, conforme as métricas definidas anteriormente. Assim, as subseções seguintes apresentam as métricas agrupadas pelo objetivo que se deseja avaliar. Essas métricas, então, são aplicadas aos dois métodos, no contexto do exemplo agendador de reuniões.

5.3.1. Objetivo 1: Corretude dos Casos de Uso Gerados

Q1) EOSCU: esta questão se refere à existência de um campo no *template*, associado ao objetivo que é satisfeito com a execução do caso de uso.

Ao observarmos o *template* de especificação do caso de uso, utilizado no método de Santander, percebemos a existência do campo "*Objetivo no Contexto*", que é preenchido com uma sentença descritiva do objetivo do caso de uso. Além disso, o próprio nome do caso de uso é preenchido como sendo uma frase curta que representa o objetivo.

No método de De la Vara, vemos que não existe um campo "explícito" que descreve o objetivo. No entanto, os campos "*Extended Task Description*" e "*Business Process*" nos permitem depreender, facilmente, o objetivo da ETD.

Q2 QVN: esta questão se refere à quantidade de validações necessárias, realizadas pelos *stakeholders*, nos artefatos gerados.

No método de Santander, depreende-se a necessidade de realizar as seguintes validações: (i) Validação do modelo SD; (ii) Validação do modelo SR; (iii) Validação do diagrama de casos de uso; (iv) Validação do *template* de especificação do caso de uso. Portanto, um mínimo de quatro validações são necessárias, a fim de garantir a corretude da especificação.

No método de De la Vara, as seguintes validações são necessárias: (i) Validação do *As-Is* BPD; (ii) Validação do *To-Be* BPD; (iii) Validação do *Labelled* BPD; (iv) Validação do *Enriched* BPD; (iv) Validação do *template* de especificação do caso de uso (ETD). Portanto, cinco validações são necessárias, a fim de garantir a corretude da especificação.

5.3.2. Objetivo 2: Completude dos Casos de Uso Gerados

Q3 QEMB: esta questão se refere à quantidade de elementos que o modelo-base, a partir do qual se obtêm os casos de uso, dispõe no modelo organizacional/de processos de negócio.

No método de Santander, contabilizamos os seguintes elementos dispostos na modelagem organizacional i*: (i) Recurso; (ii) Tarefa; (iii) Objetivo; (iv) Objetivo-soft; (v) Ligação de decomposição de tarefa; (vi) Ligação meio-fim; (vii) Ligação de contribuição; (viii) Ator e limite de ator. Portanto, tem-se oito elementos no total.

No método de De la Vara, além dos elementos típicos da modelagem BPMN, totalizando dezoito (conforme figura 2.4), contabilizamos os seguintes elementos dispostos na modelagem do *Enriched* BPD: (i) *Label*; (ii) Fluxo consecutivo. Portanto, tem-se, no total, vinte elementos.

Q4 PCTPD: esta questão se refere ao percentual de campos do *template* que são preenchidos com as diretrizes.

No método de Santander, temos: QCPD = 13 (não há diretrizes, a partir das quais seja possível preencher os campos "*Prioridade*", "*Desempenho alvo*" e "*Frequência*") e QC = 16. Portanto, temos: 13/16 = 81,25% dos campos preenchidos com o auxílio das diretrizes.

No método de De la Vara, temos: QCPD = 15 (não há diretrizes, a partir das quais seja possível preencher os campos "*Frequency*", "*Critical*" e "*Business Rules*") e QC = 18. Portanto, temos: 15/18 = 83,33 % dos campos preenchidos com o auxílio das diretrizes.

Onde: QCPD corresponde à quantidade de campos preenchidos com as diretrizes, e QC é a quantidade total de campos, obrigatórios ou opcionais, que o *template* dispõe.

Q5 QPC: esta questão se refere à quantidade de passos dos cenários, principal e secundários, gerados após a aplicação das diretrizes do método. No contexto do exemplo agendador de reuniões, percebemos que não há cenário secundário gerado por nenhum dos métodos. Portanto, a contabilização se dá apenas com os passos do cenário principal. Assim, temos:

Quantidade de passos do cenário principal (Método de Santander): 4.

Quantidade de passos do cenário principal (Método de De la Vara): 9.

Q6 PUI: esta questão se refere à possibilidade de uso de includes nas ações normais dos cenários.

No método de Santander, existe a possibilidade de uso de *include*, com diretrizes que guiam o processo de obtenção deste tipo de mecanismo no fluxo principal.

No método de De la Vara, como se utiliza o conceito de fluxo consecutivo, não se percebe o uso de *include*, nem a presença de diretrizes relacionadas.

Q7 ECTAAE: esta questão se refere à existência de um campo no *template* para descrever as ações extensivas ou alternativas ao cenário principal.

No método de Santander, o *template* utilizado possui o campo "*Extensões*", que representa os fluxos secundários.

No método de De la Vara, o *template* utilizado possui os campos "*Alternatives*" e "*Extensions*", que representam os fluxos secundários. A diferença entre fluxo alternativo e extensivo neste caso, é que um fluxo alternativo contém ações alternativas que substituem o fluxo normal. Já um fluxo extensivo contém ações de extensão ao fluxo normal, ou seja, o fluxo normal é executado, mas podem ocorrer outras ações complementares.

Q8 QCPPT: esta questão se refere à quantidade total de campos possíveis para preenchimento (obrigatórios ou opcionais), suportados pelo *template* de especificação de caso de uso.

No método de Santander, a quantidade total de campos no *template* é 16.

No método de De la Vara, a quantidade total de campos no *template* é 18.

Q9 QDGDCU: esta questão se refere à quantidade de diretrizes que o método oferece para a obtenção de um diagrama de casos de uso.

No método de Santander, existe uma diretriz (Diretriz 10) referente à obtenção do diagrama de casos de uso.

No método de De la Vara, não há diretrizes que auxiliem na obtenção de um diagrama de casos de uso.

5.3.3. Objetivo 3: Complexidade do Método

Q10 QDSA: esta questão se refere à quantidade de diretrizes e sub-diretrizes que o método dispõe para guiar o processo de especificação dos casos de uso.

No método de Santander, contabilizam-se 22 diretrizes (incluindo sub-diretrizes).

No método de De la Vara, contabilizam-se 31 diretrizes.

Q11 PCTPD: a avaliação dos métodos segundo esta métrica, já foi descrita em *Q4* (ver seção 5.3.2).

5.3.4. Objetivo 4: Alinhamento dos Requisitos Especificados com as Metas Organizacionais

Q12 EOSCU: a avaliação dos métodos segundo esta métrica, já foi descrita em *Q1* (ver seção 5.3.1).

Q13 PCUGRO: esta questão se refere ao percentual de casos de uso que possuem uma relação direta com (ou satisfaz) um objetivo da organização.

No método de Santander, para o contexto do agendador de reuniões, temos: QCURDSO = 5 e QCU = 5. Portanto, temos: 5/5 = 100% dos casos de uso gerados satisfazem algum objetivo da organização. Isto, porque todos os casos de uso possuem, explicitamente, o objetivo que se deseja alcançar com sua execução.

No método de De la Vara, para o contexto do agendador de reuniões, temos: QCURDSO = 1 e QCU = 1. Onde: QCURDSO corresponde à quantidade de casos de uso que possuem uma relação direta com (ou satisfaz) um objetivo da organização, e QCU é a quantidade total de casos de uso gerados. É importante notar que, por utilizar o conceito de fluxo consecutivo (por isso não se utiliza do mecanismo de *include*), todos os passos do cenário permanecem em uma única ETD. Portanto, temos: 1/1 = 100% dos casos de uso gerados satisfazem algum objetivo da organização.

Q14 QVN: a avaliação dos métodos segundo esta métrica, já foi descrita em *Q2* (ver seção 5.3.1).

Em resumo, a tabela 5.2 ressalta os resultados obtidos pela aplicação das métricas nos métodos de Santander e De la Vara.

Tabela 5.2 Resultados da avaliação dos métodos.

Objetivo	Questão/Métrica	Método de Santander	Método de De la Vara
Objetivo 1	*Q1/EOSCU*	SIM	SIM
Objetivo 1	*Q2/QVN*	4	5
Objetivo 2	*Q3/QEMB*	8	20
Objetivo 2	*Q4/PCTPD*	81,25%	83,33%
Objetivo 2	*Q5/QPC*	4	9
Objetivo 2	*Q6/PUI*	SIM	NÃO
Objetivo 2	*Q7/ECTAAE*	SIM	SIM
Objetivo 2	*Q8/QCPPT*	16	18
Objetivo 2	*Q9/QDGDCU*	1	0
Objetivo 3	*Q10/QDSA*	22	31
Objetivo 3	*Q11/PCTPD*	Mesmo de *Q4*	Mesmo de *Q4*
Objetivo 4	*Q12/EOSCU*	Mesmo de *Q1*	Mesmo de *Q1*
Objetivo 4	*Q13/PCUGRO*	100%	100%
Objetivo 4	*Q14/QVN*	Mesmo de *Q2*	Mesmo de *Q2*

A partir da avaliação dos métodos e dos resultados evidenciados na tabela anterior, podemos concluir alguns aspectos importantes:

1. Em ambos os métodos, é explicitado o valor que a execução do caso de uso trará ao usuário;
2. O método de De la Vara requer a produção de mais artefatos (devido aos diversos tipos de BPDs envolvidos), quando comparado com o método de Santander. Por isso, exige uma quantidade maior de validações necessárias, por parte dos *stakeholders*;
3. O método de De la Vara trabalha com modelos-base (a partir dos quais se obtêm os casos de uso) mais detalhados. Os modelos utilizados dispõem de mais elementos, de modo que maiores detalhes podem ser capturados, para a geração dos casos de uso;
4. Ambos os métodos possuem diretrizes abrangentes para guiar o processo de obtenção dos casos de uso. O percentual de campos do *template* preenchidos com as diretrizes, nos dois métodos, é bem similar;
5. Em relação à completude dos cenários, o método de De la Vara é considerado mais completo, pois as diretrizes permitem que mais passos no cenário sejam criados;

6. O método de Santander, diferentemente do método de De la Vara, oferece suporte ao reuso de passos de cenários, através do mecanismo de *include*;
7. Ambos os métodos dão suporte às descrições de ações alternativas ou extensivas ao cenário principal;
8. Ambos os métodos são bem completos em quantidade de campos nos *templates*. O *template* de De la Vara supera o utilizado por Santander com dois campos a mais;
9. O método de Santander oferece suporte à diagramatização dos casos de uso gerados, diferentemente do método de De la Vara;
10. O método de Santander é considerado menos complexo, em relação à quantidade de diretrizes a serem seguidas, quando comparado com o método de De la Vara;
11. Ambos os métodos são bem sistemáticos (ver item 4);
12. Ambos os métodos apresentam casos de uso gerados alinhados às metas organizacionais;
13. Em ambos os métodos, a presença dos casos de uso gerados para o sistema se justifica em cumprimento às metas organizacionais;
14. Ambos os métodos exigem validações para atender às expectativas dos *stakeholders*, no alcance das metas da organização (ver item 2).

A seguir, é apresentado o capítulo 6, que descreve as considerações finais desta monografia, as contribuições e limitações encontradas, bem como as direções futuras a partir deste trabalho.

CAPÍTULO 6

Conclusão

"Se você achar que está no topo, não irá fazer mais escalada"

Arnold Glasow

Este capítulo apresenta as contribuições e limitações encontradas durante a elaboração deste trabalho, as direções futuras, a partir do que aqui foi desenvolvido, bem como as considerações finais desta monografia.

6.1. Contribuições e Limitações

O uso de diferentes abordagens para a engenharia de requisitos tem incentivado, cada vez mais, a realização de trabalhos que possam contribuir para alcançar qualidade na especificação dos casos de uso, e, consequentemente, dos requisitos que compõem um SI.

Diante disto, a abordagem orientada a objetivos i* possibilita que a modelagem do ambiente organizacional, com toda sua complexidade (incluindo relacionamentos e dependências entre atores, tarefas, recursos, objetivos e objetivos-soft), possa contribuir com a especificação de casos de uso de um sistema que esteja mais alinhado aos objetivos organizacionais. Isto, porque a modelagem focaliza nos objetivos da organização, dispondo de elementos capazes de fornecer informações importantes para justificar o "porquê" da existência de cada caso de uso.

A abordagem baseada em processos de negócio, utilizando uma notação incrementada do BPMN, permite que os diversos aspectos que regem os processos de negócio da organização sejam modelados. Isto permite que o engenheiro de requisitos possa extrair informações suficientes para especificar casos de uso, em forma de ETDs. Esses casos de uso, portanto, fundamentam-se nos processos de negócio que regem a organização.

Uma contribuição deste trabalho foi permitir uma análise de dois métodos para geração de casos de uso, a saber: o método de Santander (que utiliza uma abordagem orientada a objetivos) e o método de De la Vara (que utiliza uma abordagem baseada em processos de negócio). Com isso, foi possível comparar ambos os métodos, considerando os aspectos de corretude, completude, complexidade e alinhamento dos casos de uso gerados e, consequentemente dos requisitos do sistema, com as metas da organização. Esta comparação

nos permitiu fornecer um guia aos engenheiros de requisitos, no que se refere ao melhor método a ser utilizado para obter, sistematicamente, cenários de casos de uso, de acordo com os aspectos considerados mais importantes para a eleição de um método.

Como limitação deste trabalho, podemos citar a dificuldade para encontrar modelos em i* e em BPMN fielmente equivalentes, para utilizar na aplicação dos métodos. Além disso, outra limitação encontrada foi a ausência de uma avaliação qualitativa dos métodos, impossibilitada pelo curto tempo disponível para a elaboração deste trabalho.

Consideradas as contribuições e limitações deste trabalho, a próxima seção apresenta as direções e perspectivas futuras.

6.2. Direções Futuras

Como perspectivas futuras, podemos citar: (i) a realização de uma análise qualitativa dos métodos, com a aplicação de um questionário específico, apresentado no Apêndice C, no ambiente de usuários dos dois métodos; (ii) trabalhar na validação de métodos existentes que, de forma sistemática, realizam o mapeamento de i* para BPMN e vice-versa e assim contornar a primeira das limitações apresentadas na seção anterior. Isto nos permitiria ter modelos semanticamente equivalentes, para serem aplicados nos métodos de Santander e De la Vara.

A próxima seção relata algumas considerações finais advindas da elaboração desta monografia.

6.3. Considerações Finais

Com este trabalho, o engenheiro de requisitos, bem como outros envolvidos no projeto de um sistema, podem ter uma visão mais ampla dos métodos de geração de casos de uso analisados. Além disso, com a comparação dos métodos, é possível indicar os resultados de cada um, em relação à corretude, completude, complexidade do método e alinhamento dos requisitos do sistema especificado com as metas da organização.

O desenvolvimento deste trabalho contribuiu, significativamente, para o desenvolvimento acadêmico e científico do autor, uma vez que foi possível trabalhar com diversos aspectos, como técnicas de pesquisas, identificação de problema, planejamento e metodologia científica, análise comparativa de resultados e obtenção de conclusões.

APÊNDICE A

Template de Especificação de Caso de Uso Proposto por Cockburn

Caso de Uso: <número> << o nome é um objetivo descrito com uma frase curta contendo um verbo na voz ativa >>

INFORMAÇÃO CARACTERÍSTICA

Objetivo no Contexto: <uma sentença mais longa do objetivo do caso de uso se for necessário>

Escopo: <Qual sistema está sendo considerado (por exemplo, organização ou sistema computacional)>

Nível: <um dos tipos de objetivo: objetivo de usuário, objetivo de contexto ou objetivo de subfunção>

Pré-condições: <o que é necessário que já esteja satisfeito para realizar o caso de uso>

Condição Final de Sucesso: <o que ocorre/muda após a obtenção do objetivo do caso de uso>

Condição Final de Falha: <o que ocorre/muda se o objetivo é abandonado>

Ator Primário: <o nome do papel para o ator primário, ou descrição>

CENÁRIO PRINCIPAL DE SUCESSO

<coloque aqui os passos do cenário necessários para a obtenção do objetivo >

<passo #> <descrição da ação >

EXTENSÕES

<coloque aqui as extensões, uma por vez, cada uma referenciando o passo associado no cenário principal >

<passo alterado> <condição> : <ação ou sub.caso de uso >

<passo alterado > <condição> : <ação ou sub.caso de uso >

INFORMAÇÃO RELACIONADA (opcional)

Prioridade: <Quão crítico é o caso de uso para seu sistema/organização >

Desempenho alvo: <o total de tempo que este caso de uso poderia demorar >

Freqüência: <com que freqüência espera-se que o caso de uso ocorra >

Caso de Uso Pai: <opcional, nome do caso de uso que inclui este >

Casos de Uso Subordinados: <opcional, ligações para sub.casos de uso >

Atores Secundários: <lista de outros sistemas necessários para realizar este caso de uso >

Figura A.1 *Template* de especificação de caso de uso proposto por Cockburn [20].

APÊNDICE B

Estrutura de uma ETD

A aplicação do método de De la Vara em um *Enriched* BPD, juntamente com informações ou observações dos *stakeholders*, resultam na especificação de requisitos do sistema através de uma ETD. O objetivo principal de uma ETD é especificar, de maneira mais completa, adequada e precisa, o suporte que um SI pode dar na execução das tarefas do processo de negócio de uma organização [7].

Uma ETD estende o conceito de casos de uso, agregando informações de suporte às tarefas do usuário (*Task & Support Descriptions*) e fluxos de informação (*Info Case Approaches*). Ademais, atributos de qualidade também são especificados em uma ETD, segundo os padrões da ISO 9126-1. A figura B.1 mostra as características e sub-características que compõe o padrão ISO 9126-1.

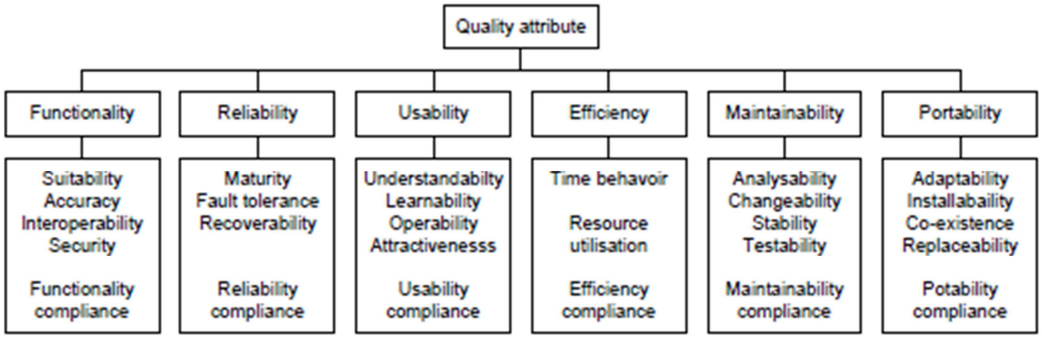

Figura B.1 Características e sub-características da ISO 9126-1 [7].

É importante mencionar que tarefas em um BPD correspondem a sub-tarefas na especificação de uma ETD. Como essas tarefas são conectadas por fluxos consecutivos em um *enriched* BPD, então é coerente afirmar que nenhuma outra ETD pode ser executada imediatamente antes ou imediatamente depois de uma ETD. Caso contrário, elas deveriam ser fundidas em um única ETD [7].

Uma ETD, basicamente, se estrutura em três partes. A primeira delas contém o nome da ETD; a segunda parte contém os requisitos do domínio (elementos próprios do domínio da aplicação e regras de negócio); a terceira e última parte contém os requisitos funcionais e não-funcionais do produto e requisitos de dados (referentes ao modelo conceitual dos dados).

As seções que compõem uma ETD são brevemente explicadas abaixo [7]:

- Nome (*name*): sentença que identifica uma ETD;
- Processo de negócio (*business process*): corresponde ao *enriched* BPD, ou seja, ao processo de negócio que é suportado pela ETD;
- Papel (*role*): representa o papel do usuário responsável pela execução da ETD, ou o sistema;
- Sub-tarefas (*subtasks*): são objetos com fluxos consecutivos do processo de negócio da ETD. Representam as tarefas do negócio que é suportado pela ETD;
- *Triggers*: são os objetos de fluxo do processo de negócio de uma ETD que precede a primeira sub-tarefa. São controlados pelo sistema e a presença deles causam a necessidade de execução da ETD;
- Pré-condições (*preconditions*): são os objetos de fluxo do processo de negócio de uma ETD que precede a primeira sub-tarefa. São controladas pelo sistema e representam condições que devem ocorrer para que a ETD seja executada;
- Pós-condições (*postconditions*): são os objetos de fluxo do processo de negócio de uma ETD que sucede a última sub-tarefa. São controladas pelo sistema e representam condições que devem ocorrer logo depois da execução da ETD;
- Freqüência (*frequency*): corresponde ao número esperado de vezes que a ETD será executada dentro de um certo período de tempo;
- Crítico (*critical*): corresponde a um período de tempo ou condição em que a ETD será executada sob condições anormais ou extremas;
- Entrada (*input*): contém as entidades de domínio que são usadas ou consumidas pelas sub-tarefas da ETD e cuja informação será armazenada pelo SI;
- Saída (*output*): contém as entidades de domínio que são modificadas ou geradas pelas sub-tarefas da ETD e cuja informação será armazenada pelo SI;
- Regras de negócio (*business rules*): correspondem às regras de negócio do processo de negócio da ETD que foram especificadas textualmente a priori e serão controladas pelo sistema. Conforme necessidade, outras regras de negócio podem ser identificadas e acrescentadas;
- Intenção do usuário (*user intention*): corresponde às ações do usuário durante a execução da ETD na interação com o sistema;

- Responsabilidade do sistema (*system responsibility*): corresponde às ações que o sistema pode realizar durante a execução da ETD;
- Fluxos de informação (*information flows*):contém as informações/dados que fazem parte ou são trocados na interação entre o usuário e o sistema. Utiliza todas as entidades de domínio de entrada e de saída;
- Atributos de qualidade (*quality attributes*): correspondem aos requisitos do produto que não foram especificados em nenhuma outra seção. São os requisitos de qualidade.

As intenções do usuário e responsabilidades do sistema podem seguir um fluxo normal, cuja execução é a esperada; ou um fluxo alternativo, que contém ações alternativas e substituem o fluxo normal; ou um fluxo de extensão, que contém ações de extensão ao fluxo normal, ou seja, o fluxo normal é executado mas podem ocorrer outras ações complementares.

Outra observação importante é que os fluxos de informação são montados a partir de uma gramática BNF (ver figura B.2). Essa gramática define como as entidades de domínio que representam as informações ou dados trocados na interação do usuário com o sistema devem ser especificados.

```
<Information flow> ::= <Input flow> | <Output flow> | <Autonomous flow>
<Input flow> ::=  → <Input data expression>
<Output flow> ::=  ← <Output data expression>
<Autonomous flow> ::=  ~ <Autonomous data expression>
<Input data expression> ::= <Domain entity> |
   <Domain entity> / <Attribute> / |
   <Input data expression> + <Input data expression> |
   <Lower limit> { <Input data expression> } <Upper limit>
<Output data expression> ::= <Domain entity> / <Attribute> / |
   <Output data expression> + <Output data expression> |
   ( <Output data expression> '|' <Output data expression> ) |
   <Lower limit> { <Output data expression> } <Upper limit> |
   [ <Output data expression> ]
<Autonomous data expression> ::= <Domain entity> / <Attribute> / |
   <Autonomous data expression> + <Autonomous data expression> |
   <Lower limit> { <Autonomous data expression> } <Upper limit>
<Attribute> ::= <Attribute name> | <Attribute> + <Attribute> |
   ( <Attribute> '|' <Attribute> ) | [ <Attribute> ]
<Domain entity> ::= <String>
<Attribute name> ::= <String>
<String> ::= <Character> | <Character><String>
<Character> ::= <Letter> | <Number> | '(' | ')' |
<Lower limit> ::= <Number>
<Upper limit> ::= <Number> | n
<Letter> ::= A | a | B | b | C | c | D | d …
<Number> ::= <Digit> | <Digit><Number>
<Digit> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
```

Figura B.2 Gramática BNF para especificação de fluxos de informação [7].

A legenda para os símbolos da gramática BNF é mostrada a seguir:

‘→’, Para entrada de informações no SI;

‘←’, Para saída de informações do SI;

‘~’, Para memória autônoma ou ações ativas do SI;

‘/ /’, Para membros;

‘+’, Para agregação;

‘(|)’, Para alternativa;

‘{ }’, Para repetição;

‘[ ]’, Para opção;

‘n’, Para indeterminado número de repetições.

APÊNDICE C

Questionário

1) A execução dos casos de uso, obtidos da aplicação do método de Santander, atende a alguma meta da organização ?

2) A execução da ETD, obtida da aplicação do método de De la Vara, atende a alguma meta da organização ?

3) A utilização do método de Santander, na prática, tem atendido às expectativas, em relação à completude das especificações obtidas de casos de uso e cenários ? Por quê ?

4) A utilização do método de De la Vara, na prática, tem atendido às expectativas, em relação à completude das especificações obtidas de casos de uso e cenários, na forma de ETD ? Por quê ?

5) A aplicação do método de Santander, na prática, foi simples de ser realizada, diante da modelagem organizacional disponível ? Por quê ?

6) A aplicação do método de De la Vara, na prática, foi simples de ser realizada, diante da modelagem de processos de negócio disponível ? Por quê ?

7) Em geral, qual dos dois métodos seria preferível pela organização, para realizar a especificação dos casos de uso e cenários ? Por quê ?

Referências Bibliográficas

[1]FALBO, R. A. **Engenharia de Requisitos: Notas de Aula**. Universidade Federal do Espírito Santo, Vitória, 2012.

[2]DE LA VARA, J. L.; FORTUNA, M. H.; SÁNCHEZ, J.; WERNER, C. M. L.; BORGES M. R. S. **A Requirements Engineering Approach for Data Modelling of Process-Aware Information Systems**. 2009.

[3]FALBO, R. A. **Engenharia de Requisitos: Notas de Aula**. Universidade Federal do Espírito Santo, Vitória. 2005.

[4]SANTANDER, V. F. A. **Integrando Modelagem Organizacional com Modelagem Funcional**, 2002. 221f. Tese (Doutorado em Ciência da Computação). Centro de Informática, Universidade Federal de Pernambuco, Recife. 2002.

[5]PAULA FILHO, W. P. **Engenharia de Software: fundamentos, métodos e padrões**. Rio de Janeiro: LTC. 2000.

[6]JACOBSON, I. et. al. **Object Oriented Software Engineering: A Use Case Driven Approach**. Workingham: Addison-Wesley. 1992.

[7]DE LA VARA, J. L. **Business Process Based Requirements Specification and Object-Oriented Conceptual Modelling of Information System**, 2011. 334p. Thesis (Doctor of Philosophy in Computer Science). Universidad Politécnica de Valencia, Spain. 2011.

[8]NOGUEIRA, A. **Casos de Uso (Cenários)**. 2006. Disponível em: http://imasters.com.br/artigo/3811/uml/casos-de-uso-cenarios/. Acesso em: 25 de junho de 2013.

[9]YU, E. **Towards Modelling and Reasoning Support for Early-phase Requirements Engineering**. Requirements Engineering, Proceedings of the Third IEEE International Symposium. 1997.

[10]BASILI, V.; CALDIERA, G.; ROMBACH, H.D. **The Goal Question Metric Approach**. 1994.

[11]TOMIYASU, L.; ITO, M. **Análise da Dificuldade de Comunicação entre o Analista de Requisitos e Usuários**. Disponível em: http://www.centropaulasouza.sp.gov.br/pos-graduacao/workshop-de-pos-graduacao-e-pesquisa/007-workshop-2012/workshop/trabalhos/desenvgestti/analise-da-dificuldade.pdf. Acesso em: 20 de agosto de 2013.

[12]KOTONYA, G.; SOMMERVILLE, I. **Requirements Engineering: processes and techniques**. Chichester, John Wiley & Sons. 1998.

[13]FREIRE, M.A. **Requisitos de Software**. Departamento de Informática. UFMA. Disponível em: http://www.deinf.ufma.br/~maria/arqan/cap3-requis.pdf. Acesso em: 18 de agosto de 2013.

[14]LAMSWEERDE, A. **Goal-Oriented Requirements Engineering: A Guided Tour**. Requirements Engineering, (RE'01). Proceedings. Fifth IEEE International Symposium. 2001.

[15]DECREUS, K.; SNOECK, M.; POELS, G. **Practical Challenges for Methods Transforming i* Goal Models into Business Process Models**. In: Seventeenth IEEE International Requirements Engineering Conference (RE'09), Atlanta, GA, USA. 2009.

[16]WESKE, M. **Business Process Management: Concepts, Languages, Architectures**. Springer, Heidelberg. 2007.

[17]PAIM, R. **As tarefas para gestão de processos**. Rio de Janeiro: Tese de Doutorado em Engenharia de Produção, UFRJ, 2007.

[18]OMG. **Business Process Model and Notation (BPMN) Specification**. 2009. v.2.0 (online) Disponível em: http://www.bpmn.org. Acesso em: 10 de agosto de 2013.

[19]ALVES, R.; SILVA, C.; CASTRO, J. **A bi-directional mapping between i* and BPMN models in the context of business process management**. In: Requirements Engineering @ Brazil. Available at: http://www.cin.ufpe.br/~erbr13/arquivos/proceedings/erbr2013_submission_33.pdf

[20]COCKBURN, A. **Writing Effective Use Cases, Humans and Technology**, Addison-Wesley. 2000.

[21]LAUESEN, S. **Software Requirements: Styles and Techniques**. Addison-Wesley, London. 2002.

[22]FORTUNA, M.H.; WERNER, C.M.L.; BORGES, M.R.S**. Info Cases: Integrating Use Cases and Domain Models**. In: 16th International Requirements Engineering Conference (RE'08), pp 81-84. 2008.

[23]BOOCH, G.; JACOBSON, I.; RUMBAUGH, J. **The Unified Modeling Language User Guide**, Addison-Wesley. 1999.

[24]ROLLAND, C.; SOUVEYET, C.; ACHOUR, C. B. "**Guiding Goal Modeling Using Scenarios**", IEEE Transactions on Software Engineering, Vol 24, No 12, Special Issue on Scenario Management, December, 1998.

[25]ANTON, A. **Goal identification and refinement in the specification of software-based information systems**. Phd Thesis, Georgia Institute of Technology, Atlanta, GA, June, 1997.

[26]KRUCHTEN, Philippe. **The Rational Unified Process**, Addison-Wesley Pub Co, ISBN: 0201707101, 2nd edition. 2000.

[27]D'SOUZA; DESMOND, F.; WILLS, A.C. **Objects, Components and Frameworks with UML: The Catalysis Approach**, Addison-Wesley, November, 1998.

[28]ROLLAND, C. **Capturing System Intentionality with Maps**. In: Krogstie, J., Opdhal, A.L., Brinkkemper, S. (eds.) Conceptual Modelling in Information Systems Engineering. Springer, Heidelberg, pp 141-158. 2007.