Pós-Graduação em Ciência da Computação

"Integrando Modelagem Organizacional com Modelagem Funcional"

Por

Victor Francisco Araya Santander

Tese de Doutorado

Universidade Federal de Pernambuco
posgraduacao@cin.ufpe.br
www.cin.ufpe.br/~posgraduacao

RECIFE, Dezembro/2002

Victor Francisco Araya Santander

Integrando Modelagem Organizacional com Modelagem Funcional

Tese apresentada à Coordenação da Pos-Graduação em Ciência da Computação do Centro de Informática, como parte dos requisitos para obtenção do título de Doutor em Ciência da Computação.

Orientador: Jaelson Freire Brelaz de Castro

Recife

Dezembro, 2002

AGRADECIMENTOS

A Deus por permitir mais esta evolução pessoal e profissional.

À minha esposa Jaqueline pelo seu apoio, paciência e carinho incondicional, os quais foram fundamentais para a realização deste trabalho. Ao meu filho Victor por ser a razão desta nova conquista.

Aos meus pais Elizabeth e Victor pelo amor, incentivo e dedicação em todos os momentos de minha vida. Aos meus irmãos Pablo, Márcia e Elizabeth pelo carinho, amizade e constante apoio.

Ao meu orientador Jaelson Castro pela amizade, solidariedade, confiança, motivação, apoio e orientação no desenvolvimento deste trabalho. Muito obrigado.

Aos membros da banca examinadora do Exame de Proposta de Tese, composta pelos professores Alexandre Marcos Lins de Vasconcelos, Francisco de Assis Cartaxo Pinheiro e Julio César Sampaio do Prado Leite que forneceram importantes comentários e sugestões para melhorar a tese.

Aos colegas Gilberto Cysneiro Filho, Fernanda Alencar e Marcio Cornélio pela amizade e constantes comentários sobre o trabalho. Aos demais colegas do Laboratório de Engenharia de Requisitos pela amizade e discussões fundamentais para a evolução do trabalho.

Resumo

Entre as principais preocupações no desenvolvimento de software, destacamos a necessidade de elicitar, compreender e especificar adequadamente os requisitos de sistemas de software. Este trabalho é realizado em conjunto por engenheiros de requisitos e usuários e/ou clientes que solicitam o software. A Engenharia de Requisitos tem apresentado algumas técnicas para auxiliar neste processo. Técnicas baseadas em cenários têm sido bastante utilizadas e recebido uma atenção especial. Cenários podem ser utilizados para descrever as interações entre usuários e sistemas de software, objetivando alcançar algo relevante para o usuário no uso do sistema. Neste contexto, Caso de Uso é um tipo de técnica baseada em cenários que tem se destacado. Isto decorre do fato, de que Casos de Uso, integram e são considerados essenciais na *UML (Unified Modeling Language)*, uma linguagem padronizada para modelagem visual, a qual tem sido considerada um dos mais importantes avanços no paradigma de desenvolvimento orientado a objetos. Tipicamente, Diagramas de CASOS DE USO têm sido usados para capturar requisitos funcionais do sistema a ser desenvolvido.

Contudo, o desenvolvimento de sistemas computacionais ocorre dentro de um contexto no qual processos organizacionais estão bem estabelecidos. Portanto, é preciso capturar os requisitos organizacionais para definir como o sistema pretendido irá satisfazer os objetivos da organização, por que ele é necessário, quais as alternativas existentes, quais as implicações das alternativas para as várias partes interessadas, etc. Lamentavelmente, UML e técnicas baseadas em cenários em geral, não estão equipadas para modelar os requisitos organizacionais. Precisamos de outra técnica, tal como i*, para representar estes aspectos. Contudo, os requisitos organizacionais precisam ser relacionados aos requisitos funcionais, representados através de Diagramas de Caso de Uso.

Neste trabalho, mostramos a viabilidade e as vantagens de integrar i* e Casos de Uso. São descritas algumas diretrizes para auxiliar engenheiros de requisitos a desenvolver Diagramas de Caso de Uso em UML a partir dos modelos organizacionais propostos na técnica i*. Aplicamos estas diretrizes ao problema bastante conhecido de Agendamento de Reuniões *(Meeting Scheduler)* e a uma aplicação de comércio eletrônico.

Palavras-Chave: Cenários, Modelagem Organizacional, Engenharia de Requisitos.

Abstract

Among the main concerns in the software development, we have detected the need to understand, elicit and specify appropriately, the software system requirements. This work is accomplished together by requirements engineers and users and/or customers that request the software. Requirement engineering has presented some techniques to assist requirements engineers in this process. Among these techniques, we can highlight scenario-based techniques. Scenarios can be used to describe the interactions between users and software systems, aiming at reaching something relevant from the user point of view. In this context, use case is a relevant and important scenario-based technique. Use Case has been integrated and considered an essential component in the Unified Modeling Language (UML), a standard for visual modelling, one of the most important advances in the object-oriented development. In UML, Use Case has been used for capturing system functional requirements. However, system development occurs in a context where organisational processes are well established. Therefore, we need to capture organisational requirements to define how the system fulfils the organisation goals, why it is necessary, what are the possible alternatives, what are the implications to the involved parts, etc. Unfortunately, UML and other scenario-based techniques are not well equipped for modeling organisational requirements. We need others techniques, such as i*, to represent these aspects. Nevertheless, organisational requirements must be related to functional requirements represented as use cases. In this work, we present the theoretical foundation that allow us to observe the viability and the advantages of the i* and scenario-based technique integration. We describe some guidelines to assist requirements engineers to develop Use Cases in UML from the organisational models in i*. We have applied these guidelines to the Meeting Scheduler's problem as well as an e-commerce example.

Key-words: Scenarios, Organisational Modeling, Requirements Engineering.

Conteúdo

Índice de Figuras

Índice de Tabelas

Capítulo 1

Introdução

Neste capítulo, situamos o nosso trabalho no âmbito da Engenharia de Requisitos e apresentamos os principais objetivos da nossa proposta. Apresentamos inicialmente na seção 1.1, as principais dificuldades atuais no desenvolvimento de software, destacando a importância do processo de Engenharia de Requisitos na obtenção de produtos de software de qualidade. Na seção 1.2, apresentamos as motivações para a realização do trabalho e na seção 1.3, expomos uma visão geral bem como os objetivos específicos da nossa proposta. Na seção 1.4, descrevemos as contribuições esperadas do trabalho e na seção 1.5, apresentamos a estrutura de apresentação da tese.

1.1 Contexto

O desenvolvimento de softwares cada vez mais complexos, passíveis de certificação e com menor custo possível, tem se tornado um desafio constante para a comunidade de Engenharia de software. Diversas técnicas, metodologias e ferramentas vêm sendo propostas com o intuito de suportar e auxiliar a produção de software de qualidade. A crise do software, bem descrita por Brooks (1987), continua em muitos aspectos vigente nos nossos dias. É comum encontrarmos softwares que não atendem adequadamente aos requisitos de usuários e/ou contendo funções não solicitadas pelos mesmos. Entre os principais motivos para este insucesso podemos destacar: a falta de processos de desenvolvimento bem definidos; requisitos de software não bem compreendidos e acordados; uso de técnicas inadequadas; bem como a própria complexidade associada com softwares a serem desenvolvidos. Alia-se a estes aspectos,

as constantes pressões externas que fazem com que engenheiros de software e demais profissionais da área sejam obrigados a entregar produtos de software, em um tempo menor que o necessário para se obter um produto de qualidade (Kauppinen et al. 2002).

Neste contexto, torna-se papel fundamental de engenheiros de software procurar alternativas que minimizem as possibilidades de se obter softwares não confiáveis, incompletos e de baixa qualidade. Assim, é necessário detectar quais são as atividades e fases críticas no processo de desenvolvimento de software e desenvolver técnicas, metodologias, processos e ferramentas que possam auxiliar neste sentido. As pesquisas atuais na Engenharia de software têm lidado com os principais problemas envolvendo estes pontos críticos. No entanto, tem existido um consenso da comunidade de Engenharia de software em apontar as etapas iniciais do desenvolvimento de software como as mais críticas e as de maior impacto sobre a qualidade do produto final (Kotonya et al. 1997) (Sommerville et al. 1997). Isto implica em afirmar que, de forma geral, independente da natureza do software a ser desenvolvido, elicitar, analisar, negociar, especificar e gerenciar adequadamente requisitos, é de fundamental importância para a obtenção de um produto de qualidade.

Dada a importância desta fase inicial de descoberta e especificação de requisitos, surgiu a *Engenharia de Requisitos*, cujo processo é definido por Kotonya e Sommerville (1997), como *"um conjunto estruturado de atividades que são seguidas para derivar, validar e manter um documento de requisitos de sistema"*. Freqüentemente, requisitos de software são inadequadamente elicitados, analisados e especificados. Solucionar problemas de incompletude e inconsistência de requisitos, ainda nas etapas iniciais do processo de desenvolvimento, possibilita-nos a obtenção de produtos de softwares mais confiáveis e que de fato atendam às reais necessidades dos clientes e usuários.

Assim, torna-se necessário impulsionar as pesquisas na Engenharia de Requisitos, procurando-se obter técnicas e metodologias que considerem todos os aspectos relevantes envolvidos na obtenção de documentos de requisitos mais completos e que representem o que o usuário espera de sistemas de software pretendidos. Elaborar documentos de requisitos envolve entre outros aspectos: considerar todos os elementos do ambiente organizacional no qual o software executará; verificar quais são os requisitos funcionais e não funcionais desejáveis; assegurar-se que estes requisitos estão o mais completo possível; bem como garantir que os requisitos especificados estejam de acordo com o que usuários solicitaram. Na próxima seção, apresentamos as principais motivações para a realização deste trabalho.

1.2 Motivações

O desenvolvimento de novas técnicas de suporte às atividades da Engenharia de Requisitos tem sido uma preocupação atual da comunidade acadêmica e industrial. Diversas abordagens para elicitar, analisar, especificar e gerenciar requisitos têm surgido desde as primeiras técnicas utilizadas com esse fim. No início, a análise estruturada de sistemas (DeMarco 1979), através de técnicas tais como DFDs (Diagramas de Fluxo de Dados), DD (Dicionários de Dados) e DER (Diagramas de Entidade e Relacionamentos), procurava atentar para a análise de dados e requisitos. Atualmente, técnicas tais como cenários, etnografia, entrevistas, prototipação, métodos formais e outras, também são apontadas como alternativas para suportar as atividades de Engenharia de Requisitos.

Dentre estas abordagens, técnicas baseadas em cenários têm recebido uma atenção especial. É importante salientar que Cenários têm sido utilizados para vários fins na Engenharia de Requisitos/software incluindo especificações de processos de

negócio, sistemas de software já existentes, bem como especificações de interações entre usuários e sistemas computacionais a serem desenvolvidos (Pohl et al. 1997). Várias abordagens de cenários têm sido desenvolvidas entre as quais podemos destacar Casos de Uso (Jacobson 1995), cenários apresentados no projeto CREWS (Ralyté 1999) e o *framework* de desenvolvimento de cenários descrito em Leite et al (1997). Casos de Uso têm-se destacado por fazerem parte e serem ponto chave na Linguagem de Modelagem Unificada (UML) (Booch et al. 1999), a qual é um padrão de linguagem de modelagem para o desenvolvimento de software orientado a objetos.

Um dos pontos positivos do uso de cenários é que os mesmos são, normalmente, facilmente compreendidos pelos diversos *stakeholders* (Breitman et al. 1998). Esta característica tem facilitado e auxiliado no trabalho de entendimento, negociação e concordância dos requisitos de sistemas de software. Cenários são utilizados não somente para a elicitação de requisitos, mas também na sua validação e modelagem (Potts 1999) (Sutcliffe et al. 2002).

Assim, uma das maiores dificuldades da Engenharia de Requisitos, que é lidar com diversos usuários e grande quantidade de informações, pode ser minimizada e contornada com o uso de cenários. Contudo, mais recentemente, cenários vêm sendo utilizados em conjunto com abordagens orientadas a objetivos (Anton 1997) (Anton et al. 2001) (Potts 1999) (Cockburn 2000) (Ralyté et al. 1999). Abordagens orientadas a objetivos têm como foco expressar os "porquês" associados com a construção de sistemas. Estes "porquês" expressam as razões e justificativas para sistemas propostos. Objetivos, sob este ponto de vista, são metas que clientes e usuários possuem em relação a sistemas de software e que podem ser descritos e refinados em cenários. Os trabalhos, integrando cenários e objetivos, defendem a idéia de que focalizando-se nos objetivos no início do processo de Engenharia de Requisitos, permite-se que analistas

comuniquem-se com os stakeholders, utilizando uma linguagem baseada nos conceitos com os quais ambos estão familiarizados e sentem-se confortáveis. Assim, a partir da descoberta inicial de objetivos de sistemas pode-se gerar e descrever cenários bem como da análise de cenários pode-se derivar outros objetivos para sistemas computacionais. Este processo é iterativo, buscando novos requisitos de sistemas.

Apesar do consenso e do reconhecimento de cenários como ferramenta importante no processo de Engenharia de Requisitos, podemos apontar algumas carências da técnica, principalmente no que diz respeito à inclusão de aspectos inerentes ao ambiente organizacional no qual o software está inserido. Quando usuários desejam desenvolver um software, em grande parte das situações, não há uma idéia concreta do que os mesmos desejam. Geralmente o que se tem, são intenções e desejos de facilitar a execução de atividades no ambiente organizacional. Por outro lado, o processo de Engenharia de Requisitos atenta inicialmente para a necessidade de elicitar os requisitos do sistema junto ao usuário, procurando obter seus desejos e expectativas sobre o sistema. Muitos dos problemas associados com o desenvolvimento de software podem iniciar nesta fase. Detectar o que é realmente relevante para o usuário, levando-se em consideração os objetivos organizacionais, não é uma tarefa trivial. Técnicas baseadas em cenários auxiliam nesta tarefa, mas precisam ser complementadas.

É bastante aceito que o trabalho da Engenharia de Requisitos pode ser melhorado se modelarmos aspectos organizacionais visando entender melhor as intenções e motivações organizacionais que incorporam o desejo do desenvolvimento de um software. Neste sentido, algumas propostas objetivando a *modelagem organizacional* têm sido apontadas (Bubenko 1993) (Yu 1995). Este tipo de modelagem objetiva fornecer recursos que permitam modelar as intenções, relacionamentos e motivações entre os membros de uma organização, bem como também descrever metas

organizacionais que possam originar e nortear o desenvolvimento de sistemas de software. A partir destes modelos, permite-se compreender melhor o funcionamento do ambiente organizacional, como também as relações humanas e de trabalho entre os participantes da organização. Com estas informações, os requisitos de uma solução computacional para processos organizacionais podem ser melhor elicitados e especificados.

Tendo em vista este panorama, motiva-nos o fato de podermos evoluir de modelos organizacionais, envolvendo o domínio de um sistema computacional a ser desenvolvido, para uma elicitação e especificação de requisitos mais completa e consistente, utilizando-se a técnica baseada em cenários denominada **caso de uso**. É importante também salientar que adotamos neste trabalho o modelo de escrita de casos de uso proposto por Cockburn (2000), no qual cada caso de uso possui um objetivo associado que representa o que o usuário espera obter como resultado da execução do caso de uso. Os objetivos e escopo da nossa tese são descritos na próxima seção.

1.3 Proposta

Nesta tese, descrevemos um conjunto de diretrizes visando indicar a viabilidade e as vantagens de se integrar no processo de Engenharia de Requisitos, modelos organizacionais desenvolvidos através do *framework* i*, com técnicas baseadas em cenários, representadas nesta tese pela técnica denominada de Caso de Uso. Um dos motivos da escolha do *framework* i* para modelar ambientes organizacionais é que o mesmo é um dos pilares de um projeto mais abrangente de desenvolvimento de software orientado a requisitos/objetivos denominado *Tropos* (Mylopoulos et al. 2000). Este projeto argumenta que a compreensão inicial do problema a ser resolvido na organização, bem como de que forma sistemas computacionais podem colaborar na

solução deste problema, é um aspecto essencial no processo de Engenharia de Requisitos. A técnica i* propicia esta compreensão através de mecanismos que permitem expressar tarefas, objetivos, objetivos-soft e recursos associados com intenções e necessidades de atores no ambiente organizacional. Em contraste a outras técnicas de modelagem organizacional tais como fluxos de trabalho (*workflows*) *)* (Bortoli et al. 2000) (Estrada et al. 2001) (Penadés et al. 2001) e regras de negócio (Leonardi et al. 1998) (Rosca et al. 1997), i* permite expressar além da descrição das atividades de processos de negócio, os aspectos intencionais, motivações e possíveis alternativas a serem consideradas em processos de negócio apoiados por sistemas computacionais. Além desta principal motivação, i* também foi escolhida por ser de fácil entendimento pelos diversos stakeholders, utilizando conceitos próximos do domínio do conhecimento de atores organizacionais.

Assim, os modelos organizacionais desenvolvidos através da técnica i*, permitem analisar a organização e seus processos de negócio, bem como possíveis impactos que sistemas computacionais podem provocar em ambientes organizacionais. Com base nesta análise, os esforços podem ser direcionados na captura da funcionalidade destes sistemas, adotando técnicas tais como, casos de uso.

É importante salientar que a técnica de Caso de Uso é um pilar do paradigma de orientação a objetos amplamente utilizado tanto no meio acadêmico, quanto industrial. Este fato e a dificuldade detectada por engenheiros de requisitos e desenvolvedores na elicitação e descrição de casos de uso (Cockburn 2000) (Schneider 1998) (Lilly 1999) é uma das fortes motivações para derivar casos de uso a partir de modelos organizacionais. Defendemos a idéia de que i* é mais adequada para modelar ambientes organizacionais, principalmente por lidar com questões associadas com aspectos intencionais e de motivações na organização. Estes modelos tornam-se uma fonte de

conhecimento do ambiente organizacional acordada por todos os *stakeholders* e podem também ser utilizados para derivar casos de uso através do uso de um conjunto de diretrizes propostas nesta tese. Os casos de uso gerados tendem a ser mais estáveis devido às interações prévias realizadas entre os *stakeholders* para construir modelos organizacionais, nas quais estabelece-se um consenso do que deve ser considerado, principalmente, do ponto de vista de metas e tarefas associadas com sistemas computacionais pretendidos.

Outro aspecto também importante de nossa pesquisa está no estudo da Engenharia de Requisitos orientada a objetivos (*GORE – Goal Oriented Requirements Engineering*). Sob este enfoque da Engenharia de Requisitos (Anton 1997) (Potts 1997) (Dardene et al. 1993) (Mylopoulos et al. 2000), a captura de requisitos dar-se-á através da elicitacão e operacionalização de objetivos em relação a sistemas computacionais. Utilizando este enfoque, estamos propondo algumas diretrizes para derivar a partir de modelos organizacionais em i*, os objetivos de sistemas computacionais descritos como objetivos de casos de uso. Guiar o processo de desenvolvimento de software, com base em objetivos, significa definir quais são os objetivos de usuários e da organização em relação ao sistema desejado. Objetivos de alto nível da organização podem ser refinados, originando outros objetivos de mais baixo nível, os quais podem ser operacionalizados para integrarem sistemas computacionais. Este processo de refinamento e operacionalização normalmente recai na descoberta de requisitos funcionais e não funcionais do sistema. Contudo, este processo não é trivial, pois depende da correta compreensão das necessidades dos usuários, bem como de um constante trabalho de interação, análise e negociação entre todos os *stakeholders*.

Desta forma, argumentamos que através da modelagem organizacional realizada com a técnica i*, intenções, motivações e objetivos organizacionais podem ser

representados e posteriormente mapeados/integrados com casos de uso. Este processo de mapeamento/integração envolve a análise dos objetivos (*goals*) e dos elementos do tipo tarefa, recurso e "objetivo-soft" representados em i*. Objetivos (*Goals*) no *framework* de i* (Yu 1995), são representados tipicamente como dependências entre atores da organização. Um ator depende de outro ator para alcançar um objetivo (*goal*) em um contexto de processos organizacionais. A satisfação destes objetivos (*goals*) pode incluir a realização de tarefas, bem como obtenção/disponibilidade de recursos. Com base na análise destes objetivos (*goals*), e refinamentos dos mesmos, quando necessário, casos de uso podem ser descobertos e o processo de Engenharia de Requisitos pode ser conduzido de forma mais orientada e sistemática. Assim, a partir de objetivos (*goals*) em i*, bem como também dos relacionamentos entre atores do tipo tarefa, recurso e "objetivo-soft", pode-se reconhecer uma série de requisitos funcionais e não funcionais associados diretamente com sistemas computacionais pretendidos pela organização. A figura 1 mostra uma visão geral do processo de mapeamento/integração de i* com caso de uso.

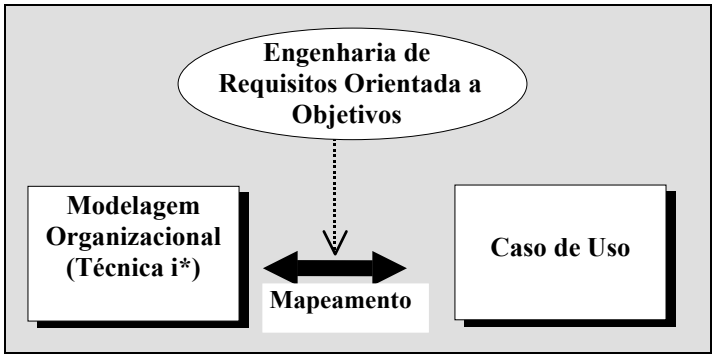

Figura 1. Uma visão do processo de mapeamento entre modelos organizacionais e casos de uso.

O mapeamento e integração entre modelagem organizacional e caso de uso possibilita um melhor rastreamento dos requisitos ao longo de todo o processo de Engenharia de Requisitos. Obtém-se também com esta integração, uma maior garantia

de que cenários representam ou estão associados a requisitos e objetivos relevantes da organização e que posteriormente serão implementados em um sistema de software.

Outra vantagem desta abordagem, é o maior compromisso consciente entre clientes e engenheiros de requisitos em relação ao documento de requisitos, por compreenderem tanto o ambiente organizacional, quanto o processo de mapeamento de requisitos, partindo de modelos organizacionais para casos de uso.

Assim, embora existam várias propostas na literatura para a descrição de cenários (Pohl et al. 1997), nesta tese, apresentaremos de forma mais específica algumas diretrizes que se aplicam à integração de modelos organizacionais desenvolvidos através do framework i*, com cenários descritos através de caso de uso. A técnica de caso de uso foi escolhida por ser parte essencial da UML, como também por ser amplamente utilizada tanto na comunidade acadêmica como na industrial. Uma descrição dos aspectos básicos de outras técnicas baseadas em cenários, bem como das diferenças entre estas técnicas, pode ser encontrada em Pohl et al. (1997).

Assim, objetivamos neste trabalho, estabelecer uma correlação entre os elementos em ***i**** e os elementos componentes de cenários sob a forma de ***Caso de Uso*****.** A partir desta correlação, fatores do ambiente organizacional inicialmente não considerados em Casos de Uso poderão ser integrados aos mesmos. Versões prévias deste trabalho podem ser encontradas em (Santander et al. 2001) (Santander et al. 2002) (Santander et al. 2002a). Na próxima seção, apresentamos as contribuições esperadas com a realização deste trabalho.

1.4 Contribuições Esperadas

Consideramos benéfico e essencial para a Engenharia de Requisitos o fato de poder obter uma compreensão inicial dos relacionamentos organizacionais em domínios

de negócio através de técnicas tais como i* (Yu 1995) e *tropos* (Mylopoulos et al. 2000). Tipicamente estas técnicas provêem um suporte para a fase inicial da Engenharia de Requisitos ("*early requirements*"), incluindo atividades que consideram, por exemplo, como o sistema pretendido irá satisfazer os objetivos da organização, por que ele é necessário, quais as alternativas existentes, quais as implicações das alternativas para as várias partes interessadas, etc. No entanto, à medida que prosseguimos no processo de Engenharia de Requisitos, precisamos nos concentrar nos requisitos funcionais e não funcionais do sistema a ser desenvolvido, o qual suportará a alternativa escolhida entre aquelas consideradas durante a fase de estudo dos requisitos iniciais ("*early requirements*"). A definição destes requisitos para o sistema pretendido ocorre na fase denominada de *"late requirements"* da Engenharia de Requisitos resultando na especificação detalhada e precisa de "o que" o sistema deverá fazer. Neste trabalho, adotamos a técnica de Caso de Uso para descrever os requisitos funcionais do sistema pretendido. Neste aspecto, argumentamos que o desenvolvimento de casos de uso a partir de modelos em i*, permite que engenheiros de requisitos possam estabelecer uma relação mais clara entre os requisitos funcionais de sistemas pretendidos e os objetivos organizacionais previamente definidos na modelagem organizacional. Além disso, utilizando os conceitos provenientes da Engenharia de Requisitos orientada a objetivos (*GORE – Goal Oriented Requirements Engineering*), podemos analisar os modelos organizacionais em i*, buscando derivar e mapear objetivos, tarefas, "objetivos-soft" e recursos para casos de uso. Destacamos que além dos objetivos de casos de uso, podemos também derivar a partir dos modelos em i*, os passos dos cenários (também conhecidos como fluxo de eventos), para os casos de uso descobertos. De forma mais detalhada, podemos apontar os seguintes benefícios e contribuições esperadas com a nossa proposta:

- entendimento das razões ou "porquês": vários pesquisadores (Anton 1997) (Castro et al. 2001) (Castro et al. 2001a) (Chung et al. 2000) (Dardene et al. 1993) (Rolland et al. 1998) (Yu 1995) (Yu et al. 1998), têm considerado objetivos em várias diferentes áreas da Engenharia de Requisitos. Abordagens orientadas a objetivos utilizadas para elicitação e descrição de requisitos podem ser contrastadas com técnicas que tratam requisitos como consistindo apenas de processos e dados, como ocorre na tradicional análise estruturada de sistemas ou "objetos", como ocorre na conhecida análise orientada a objetos. Tipicamente estas técnicas tradicionais não capturam explicitamente os "porquês" e "comos" em termos de objetivos. Entender principalmente os "porquês", que expressam as razões e justificativas para sistemas computacionais propostos, é considerado atualmente um aspecto essencial para o sucesso na elaboração de documentos de requisitos destes sistemas.

- melhoria da avaliação da dinâmica dos processos de negócio: os relacionamentos entre sistemas e seus ambientes podem também ser expressos em termos de relacionamentos baseados em objetivos. Isto é, parcialmente motivado pelos atuais ambientes organizacionais e de negócio cada vez mais dinâmicos, nos quais sistemas computacionais são crescentemente utilizados para, fundamentalmente, modificar processos de negócio (Yu et al. 1998). Derivar casos de uso, a partir dos relacionamentos em i*, permite que possamos rastrear e avaliar os impactos que as mudanças nas organizações (principalmente mudanças nos objetivos em relação a sistemas pretendidos) causam nos requisitos funcionais e não funcionais dos

sistemas computacionais. Tipicamente, uma mudança nos processos de negócio em um ambiente organizacional modifica ou cria novos relacionamentos do tipo tarefa, recurso, objetivo ou objetivo-soft entre atores nos modelos organizacionais em i*. Como o nosso processo de derivação de casos de uso toma como base estes elementos, podemos então analisar se determinados relacionamentos (tarefa, objetivo, recurso, objetivo-soft), previamente mapeados para casos de uso, foram modificados em virtude de alterações nos processos organizacionais e como estas mudanças afetam os casos de uso derivados para sistemas computacionais pretendidos.

- melhoria no processo de desenvolvimento de casos de uso: alguns dos principais problemas e deficiências associados com o processo de desenvolvimento de casos de uso (Lilly 1999), podem ser parcialmente solucionados utilizando a nossa proposta. Por exemplo, casos de uso na nossa abordagem são exatamente escritos do ponto de vista dos atores do sistema (e não do ponto de vista do sistema como ocorre, erroneamente, algumas vezes). Outro aspecto positivo é a habilidade de poder definir inicialmente os casos de uso essenciais para o sistema pretendido. Isto evita definir inicialmente muitos casos de uso e permite gerenciar a granularidade apropriada de suas descrições. Além disso, a integração de engenheiros de requisitos e clientes durante o desenvolvimento dos modelos organizacionais, contribui para que clientes (atores) compreendam melhor os casos de uso originados a partir destes modelos. Uma descrição mais detalhada dos problemas com casos de uso apresentados em Lilly (1999),

bem como possíveis soluções adotando-se à nossa proposta, são descritas no capítulo 5.

- melhoria na descrição dos requisitos: elicitar e especificar requisitos de sistemas computacionais, observando os objetivos de atores em relação ao sistema a ser desenvolvido, é uma forma de tornar requisitos mais claros (Yu et al. 1998). A partir de i* podemos derivar estes objetivos, associá-los com atores do sistema e então operacionalizá-los e refiná-los em requisitos descritos em casos de uso.

1.5 Estrutura do Trabalho

Este trabalho está dividido em 7 capítulos. No capítulo 2, apresentamos um relato da Engenharia de Requisitos, destacando os principais conceitos e pesquisas atualmente realizadas. No capítulo 3, descrevemos os principais conceitos associados com técnicas baseadas em cenários integradas com abordagens orientadas a objetivos na Engenharia de Requisitos. Neste capítulo, daremos ênfase à descrição da técnica de caso de uso mostrando a sua estrutura básica em UML, bem como as propostas de melhoria da técnica apresentadas por Cockburn (2000). No capítulo 4, descrevemos alguns aspectos relacionados à modelagem organizacional, direcionando os estudos para a técnica ***i****. No capítulo 5, mostramos a viabilidade da nossa proposta de derivação de casos de uso a partir dos modelos organizacionais em i*. Em particular, são descritas as diretrizes que podem guiar o processo de derivação. No capítulo 6, visando um melhor entendimento da proposta, as diretrizes descritas são aplicadas a dois estudos de caso: um sistema de comércio eletrônico na WEB e o problema bastante conhecido de Agendamento de Reuniões (*Meeting Scheduler*). No capítulo 7, são apresentadas as considerações finais do trabalho, bem como possíveis trabalhos futuros.

Capítulo 2

Engenharia de Requisitos

A Engenharia de Requisitos tem sido reconhecida como uma das mais importantes fases do processo de Engenharia de software. Este reconhecimento decorre da descoberta que a maior parte dos problemas e geralmente os mais dispendiosos e de maior impacto negativo no desenvolvimento de software, são originados nas etapas iniciais do desenvolvimento. Estas etapas constituem o processo de Engenharia de Requisitos, no qual, as principais atividades podem ser definidas como: elicitação, análise, negociação, especificação, gerenciamento e validação de requisitos (Kotonya et al. 1997). Normalmente, falhas na realização destas atividades resultam em documentos de requisitos inconsistentes, incompletos e conseqüentemente, produtos de software de baixa qualidade.

Na nossa proposta de integração de modelos organizacionais e cenários, salientamos que o nosso maior interesse está voltado para as atividades de *elicitação, análise e documentação de requisitos*. Acreditamos, que a partir da observação e mapeamento dos objetivos organizacionais e demais informações previamente definidas nos modelos organizacionais em i*, possamos elicitar e descrever requisitos através de cenários representados por casos de uso, de forma mais sistemática e organizada. Iniciar o processo de Engenharia de Requisitos, tendo informações sobre relacionamentos entre os atores da organização, motivações associadas com estes atores, objetivos organizacionais, bem como expectativas sobre sistemas computacionais, faz com que

stakeholders[1] em geral, tomem decisões mais conscientes a respeito da definição de requisitos para sistemas pretendidos.

Neste capítulo, apresentamos uma descrição dos elementos mais importantes da Engenharia de Requisitos, incluindo as áreas de interesse para a nossa proposta. Inicialmente, na seção 2.1, apresentamos uma visão geral da Engenharia de Requisitos, apontando várias definições para os termos Engenharia de Requisitos e Requisitos. Na seção 2.2, fazemos uma breve discussão de como requisitos podem ser classificados. Na seção 2.3, apresentamos uma visão geral do processo de Engenharia de Requisitos Na seção 2.4, são descritas as principais atividades desse processo e na seção 2.5, descrevemos as principais preocupações e tendências de pesquisa da comunidade de Engenharia de Requisitos. Na seção 2.6, são apresentadas as considerações finais do capítulo.

2.1 A Engenharia de Requisitos – Uma Visão Geral

Por se tratar de uma área de pesquisa relativamente recente na literatura, podemos encontrar várias definições da Engenharia de Requisitos. A seguir faremos uma revisão das principais definições.

A Engenharia de Requisitos é a fase do desenvolvimento de sistemas de software responsável pela identificação dos objetivos do sistema pretendido, pela operacionalização de tais objetivos em serviços e restrições e pela atribuição da responsabilidade dos requisitos resultantes para agentes como: humanos, hardware, e software (Lamswerdee 2000).

Uma outra definição é dada pelo IEEE (IEEE 1984), segundo a qual a Engenharia de Requisitos corresponde ao processo de aquisição, refinamento e

[1] Todos aqueles que interagem de alguma forma com o sistema sendo desenvolvido.

verificação das necessidades do cliente para um sistema de software, objetivando-se ter uma especificação completa e correta dos requisitos de software.

Para o projeto STARTS, segundo Hofman (1993), Engenharia de Requisitos compreende o processo pelo qual as intenções e requisitos escritos e falados de seu possuidor são transformados em uma especificação precisa, consistente, não ambígua e completa do comportamento do sistema, incluindo funções, interfaces, desempenho e limitações.

Para Leite (1987), a Engenharia de Requisitos pode ser definida como um processo, segundo diferentes pontos de vista, no qual "o que" é para fazer é capturado e modelado. Neste processo, utiliza-se uma combinação de métodos, ferramentas e atores, sendo produzido, como resultado da modelagem, um documento de requisitos.

Boehm (1989) define Engenharia de Requisitos como uma atividade que objetiva desenvolver uma especificação completa, consistente, não ambígua e correta dos requisitos, que sirva, inclusive, de base para um acordo entre as partes envolvidas no processo de desenvolvimento do software, onde se pactue, de forma concisa, "o que" o produto irá fazer. Nesse sentido, a Engenharia de Requisitos caracteriza-se como um processo que requer um envolvimento muito grande entre cliente e desenvolvedores, evitando decisões de projetos durante a definição dos requisitos, ficando assim, definidas, pelo menos, duas fases bem distintas: o que se tenta alcançar e como projetar o sistema para satisfazer os requisitos. Segundo Hofman, essa definição parece separar a Engenharia de Requisitos de outras preocupações do desenvolvimento de software (Hofman 1993).

Já segundo Davis (1993), a Engenharia de Requisitos corresponde à atividade de entendimento das necessidades do usuário no contexto do problema a ser resolvido, bem como das limitações impostas na solução.

De acordo com Loucopoulos et al. (1995), a Engenharia de Requisitos corresponde ao processo sistemático de desenvolvimento de requisitos, através de um processo iterativo, cooperativo de análise do problema, documentação das observações resultantes, em uma variedade de formatos de representações e checagem da acurácia da compreensão ganha.

Segundo Kotonya et al. (1997), Engenharia de Requisitos trata-se de um termo relativamente novo, para cobrir todas as atividades envolvidas na descoberta, documentação e manutenção de um conjunto de requisitos, num sistema baseado em computador. Nesta definição, o termo Engenharia implica em que técnicas sistemáticas e repetitivas são usadas para assegurar que os requisitos do sistema sejam completos, consistentes e relevantes.

Para Zave (1997), a Engenharia de Requisitos está relacionada com a identificação de metas para serem atingidas pelo sistema a ser desenvolvido, assim como a operacionalização de tais metas em serviços e restrições.

Outra definição necessária para o correto entendimento de nosso trabalho é a de requisitos. Várias definições têm sido usadas para requisitos de software. Elas representam perspectivas diferentes e graus variados de detalhes e precisão. A IEEE (IEEE 1997) define requisito como sendo:

1) uma condição ou uma capacidade de que o usuário necessita para solucionar um problema ou alcançar um objetivo.
2) uma condição ou uma capacidade que deve ser alcançada ou possuída por um sistema ou componente do sistema para satisfazer um contrato, um padrão, uma especificação ou outros documentos impostos formalmente.
3) uma representação documentada de uma condição ou capacidade, conforme os itens (1) e (2).

A definição da IEEE cobre a visão do usuário sobre requisitos (comportamento externo do sistema), a visão do desenvolvedor (2) e o conceito fundamental que requisitos devem ser documentados (3).

Uma outra definição sugere que um requisito é uma necessidade do usuário ou uma característica, função ou atributo necessário do sistema que pode ser percebido de uma posição externa daquele sistema (Davis 1993). Essa definição enfatiza "o que" o sistema deve conter.

Segundo Kotonya et al. (1997), um requisito pode descrever:

- uma facilidade no nível do usuário; por exemplo, um corretor de gramática e ortografia.
- uma propriedade muito geral do sistema; por exemplo, o sigilo de informações não autorizadas.
- uma restrição específica no sistema; por exemplo, o tempo de varredura de um sensor.
- uma restrição no desenvolvimento do sistema; por exemplo: a linguagem que deverá ser utilizada para o desenvolvimento do sistema.

Uma definição bastante simples para requisitos é dada por Macaulay (1996). Segundo este autor, requisito é simplesmente algo que o cliente necessita. Já segundo Jackson (1995), requisitos são fenômenos ou propriedades do domínio da aplicação que devem ser executados, normalmente expressos em linguagem natural, diagrama informal ou em notação apropriada ao entendimento do cliente e da equipe de desenvolvimento.

2.2 Classificação de Requisitos

Durante o processo de especificação dos requisitos, surge também a necessidade de estabelecer o tipo de requisito de que se está tratando, a fim de melhorar a compreensão das necessidades do cliente, bem como modelar melhor esta necessidade.

De forma geral, podemos categorizar os requisitos em três classes básicas distintas, mas que podem estar relacionadas: funcionais, não-funcionais e organizacionais (Loucopoulos et al. 1995) (Kotonya et al. 1997) (Davis 1993). Alguns autores preferem classificar os requisitos somente segundo os dois primeiros tipos, funcionais e não-funcionais.

Os *requisitos funcionais* dizem respeito à definição das funções que um sistema ou um componente de sistema deve fazer. Eles descrevem as transformações a serem realizadas nas entradas de um sistema ou em um de seus componentes, a fim de que se produzam saídas.

Os *requisitos não-funcionais* dizem respeito a restrições, aspectos de desempenho, interfaces com o usuário, confiabilidade, segurança, manutenibilidade, portabilidade, padrões e outras propriedades que o sistema deve possuir, bem como aspectos sociais e políticos. Alguns desses requisitos são provavelmente traduzidos em funções (operacionalizados), ao longo do processo de desenvolvimento de software (Chung et al. 2000).

De forma geral, a diferença entre requisitos funcionais e não-funcionais está no fato dos primeiros descreverem “o quê” o sistema deve fazer, enquanto que os outros fixam restrições sobre “como” os requisitos funcionais serão implementados.

Os *requisitos organizacionais* dizem respeito às metas da empresa, suas políticas estratégicas adotadas, os relacionamentos entre os seus atores junto com seus respectivos objetivos.

Encontram-se ainda sugestões de que requisitos possam ser vistos como sendo "o quê" o sistema deve fazer, associado com "o porquê" deva fazer, em lugar do "como" o sistema deve fazer. Procura-se ampliar a visão tradicional dos requisitos que trata apenas de aspectos funcionais e não-funcionais, com informações organizacionais que abordam a intencionalidade dos fatos, ou seja, os requisitos organizacionais (Loucopoulos et al. 1995) (Yu 1995) (Yu 1997) (Yu et al. 1995) (Mylopoulos 1995).

Segundo essa nova visão, a Engenharia de Requisitos pode ser dita como: *A área do conhecimento preocupada na comunicação com agentes organizacionais, com respeito a suas visões, intenções e atividades relativas às suas necessidades de suporte de computadores, desenvolvimento e manutenção de uma especificação de requisitos adequada a um sistema* (Loucopoulos et al. 1995). Isso sugere que a Engenharia de Requisitos também inclua problemas e aspectos gerenciais, organizacionais, econômicos, técnicos, sociais e ambientais. Nesse contexto, em Bubenko (1995), salienta-se que a própria definição da Engenharia de Requisitos evoluiu ao longo dos anos.

A visão tradicional (Loucopoulos et al. 1995), em geral, esquece aspectos importantes, que influenciam diretamente no sistema que se deseja especificar, tais como:

- os objetivos do próprio sistema e o relacionamento desses com os objetivos da organização;
- outras propriedades dos sistemas, como as relacionadas com o desempenho, segurança e restrições.

Esta visão mais ampla sobre os requisitos, serve para caracterizar que deve ser levado em consideração, quando da especificação dos requisitos, o fato de que o sistema

que está sendo especificado é apenas um dentre os vários sistemas no ambiente da organização.

Os requisitos organizacionais desempenham papel fundamental no projeto e no desenvolvimento de um sistema. Para Dobson et al. (1994), requisitos organizacionais são aqueles que resultam do sistema pretendido, considerando-se o contexto social no qual ele se encontra inserido. Por exemplo, obrigações e responsabilidades, controle e autonomia, valores e éticas que são normalmente, embutidos na estrutura e na política da empresa, de forma que não são diretamente observados ou facilmente articulados.

A razão principal para se considerar os requisitos organizacionais, está na ajuda à compreensão de interações complexas entre as empresas (organizações) e as pessoas envolvidas. Assim, esta ajuda torna-se essencial, por oferecer meios:

- Para a descrição dos processos da organização.
- Para a definição e estruturação da informação de suporte a estas aplicações.

Embora as muitas classificações propostas tenham muitos pontos em comum, elas diferem de acordo com o ponto de vista dos autores, visto que uma dada classificação pode enfatizar mais um aspecto do que outro. Todavia, a tendência atual aponta para a necessidade de tratar, especificamente, dos requisitos organizacionais (Dardene et al. 1993) (Yu 1995) (Mylopulos 1995) (Mylopulos et al. 2000) (Castro et al. 2001).

Até pouco tempo, os projetistas e desenvolvedores ignoravam a importância dos aspectos organizacionais para o desenvolvimento de seus produtos. Só mais recentemente é que esses aspectos começaram a serem observados e foram considerados essenciais ao sucesso das implementações. Hoje, existem modelos que representam componentes intencionais, tais como: razões, motivações e os porquês.

2.3 O Processo de Engenharia de Requisitos

É consensual que de uma completa e consistente especificação de requisitos depende a qualidade do produto de software bem como uma maior satisfação do cliente. Se os requisitos são especificados de forma incompleta ou inconsistente, os artefatos resultantes das próximas etapas do processo de software (Projeto, Implementação e Testes) não terão a qualidade desejada. Quanto mais tarde problemas com requisitos forem detectados no processo de desenvolvimento, maior será o custo para corrigi-los.

Para entender melhor o processo de Engenharia de Requisitos, vejamos a figura 2, extraída de Kotonya et al. (1997).

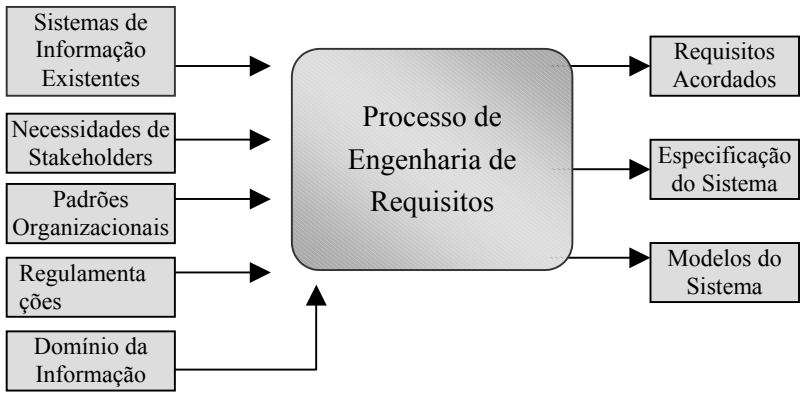

Figura 2. Entradas e Saídas do Processo de Engenharia de Requisitos, segundo Kotonya et al. (1997).

Verificamos que as entradas para o processo de Engenharia de Requisitos incluem: *informações sobre sistemas já existentes; necessidades dos stakeholders; padrões da organização; regulamentações; e informações do domínio da aplicação*. Todas estas informações são utilizadas para a realização das atividades do processo. Como resultado (saída), obtemos os *requisitos acordados, uma especificação de requisitos e modelos do sistema*. Esta visão macro ressalta que para a realização das atividades do processo de Engenharia de Requisitos, fatores humanos e técnicos têm de

ser adequadamente tratados, objetivando desta forma, que cada resultado do processo, seja o mais completo e consistente possível.

O processo de Engenharia de Requisitos é composto basicamente pelas seguintes atividades:

- elicitação de requisitos;
- análise e negociação de requisitos;
- documentação de requisitos;
- validação de requisitos;
- gerenciamento de requisitos.

Estas atividades, não seguem rigorosamente uma certa seqüência de realização no processo de Engenharia de Requisitos. Tipicamente inicia-se com a elicitação de requisitos, juntamente com uma análise dos requisitos seguida de negociações. Estas negociações são, às vezes, necessárias, tendo em vista vários pontos de vista de usuários, os quais podem diferir em relação à importância, necessidade e/ou prioridade dos requisitos sendo definidos. Posteriormente, os requisitos então são documentados e validados. Paralelamente a todas estas atividades é realizado o gerenciamento de requisitos, no qual objetiva-se controlar a evolução e mudanças nos requisitos bem como possibilitar o rastreamento[2] dos mesmos ao longo de todo o processo de desenvolvimento.

De forma geral, o que na prática ocorre é uma intensa interação entre as atividades não sendo necessário concluir totalmente uma atividade para iniciar a outra.

[2] Define-se por rastreamento em Engenharia de Requisitos (Toranzo 2002), a possibilidade de rastrear para frente e para trás os requisitos de um sistema. Exemplificando: **para trás,** rastreando um requisito para a fonte (quem o solicitou) ou **para frente**, rastreando um requisito para os artefatos da fase de projeto ou implementação, por exemplo

Os problemas maiores surgem na execução destas atividades. Diversos fatores dificultam o desenvolvimento de documentos de requisitos que realmente satisfaçam os usuários. Alguns problemas com requisitos, encontrados durante o processo de Engenharia de Requisitos, são relatados em Kotonya et al. (1997):

- os requisitos não refletem as reais necessidades do cliente em relação ao sistema a ser desenvolvido;
- os requisitos são inconsistentes e/ou incompletos;
- é dispendioso fazer mudanças após os requisitos terem sido acordados entre as partes (cliente e equipe de desenvolvimento);
- é comum ocorrer interpretação errônea entre clientes e equipe de desenvolvimento.

Desta forma, para atender adequadamente aos requisitos dos usuários e clientes, é necessário que as atividades do processo de Engenharia de Requisitos sejam realizadas de forma mais sistemática possível e com um suporte computacional adequado. Isto leva a processos com maior maturidade, o que permite que uma organização obtenha softwares com qualidade, não por dependência de esforços individuais, mas por uma própria evolução do seu processo, o qual utiliza boas práticas de Engenharia de Requisitos.

2.4 Atividades da Engenharia de Requisitos

Nesta seção, descrevemos as atividades do processo de Engenharia de Requisitos separadamente, apontando técnicas bem como dificuldades encontradas para a realização das mesmas. Vários relatos na literatura têm apontado algumas diferenças de nomenclatura ou seqüência de realização destas atividades. No entanto, consideramos

que de uma forma geral, as definições apresentadas em Kotonya et al. (1997), fornecem-nos uma visão adequada destas atividades.

2.4.1 Elicitação, Análise e Negociação de Requisitos

Quando desejamos solucionar um problema através de um sistema computacional, o primeiro passo deve ser o de descobrir quais são os requisitos desejáveis em relação a esse sistema. Neste processo de descoberta, o elemento essencial e mais importante é o cliente/usuário que requisita e/ou utilizará o sistema. As maiores dificuldades que surgem não são computacionais, mas de comunicação, pois o objetivo é extrair do usuário o que ele espera do sistema a ser desenvolvido. Recai no engenheiro de requisitos a tarefa de discernir o que é relevante entre as informações dadas pelo usuário, bem como lidar adequadamente com as declarações vagas do usuário a respeito do que o mesmo espera de sistemas computacionais.

Assim, a tarefa de **elicitar** os requisitos não é trivial. Também é necessário **analisar** os requisitos em relação a inconsistências e incompletudes, bem como **negociar**, solucionando conflitos, de forma que um conjunto de requisitos sejam acordados. Estas atividades são realizadas, na maioria das vezes, paralelamente e/ou de forma intercalada. O objetivo é delimitar um conjunto de requisitos que atendam os desejos dos usuários.

Entre as várias técnicas para elicitar requisitos podemos apontar: entrevistas; cenários (Hadad et al. 1999) (Haumer et al. 1998) (Leite et al. 1997) (Breitman et al. 1998); observação e análise social (Goguen et al. 1993); etnografia (Sommerville et al. 1999); entre outras. Toda técnica de elicitação deve descrever os requisitos através de uma representação facilmente entendida por todos os Stakeholders. Este é um aspecto

essencial, pois o cliente/usuário deve compreender e concordar com os requisitos que estão sendo definidos.

Após a atividade de elicitação ou mais comumente, de forma intercalada, os requisitos devem ser analisados para detectar incompletudes, omissões ou redundâncias. Na análise, a preocupação está em descobrir os requisitos que realmente são necessários e que o *stakeholder* deseja. Várias técnicas podem ser utilizadas para este fim. Entre elas destacam-se:

1. lista de Checagem da Análise: é uma lista de questões, as quais o analista pode usar para avaliar cada requisito. Cada item serve de guia na avaliação do requisito. No final da checagem, pode obter-se uma lista de problemas associados com requisitos. Estes problemas podem ser solucionados através de negociação ou se necessário, com nova elicitação.
2. matrizes de Interação: é utilizada para descobrir as interações entre requisitos, apontando possíveis conflitos entre requisitos. A atividade de negociação pode corrigir estes conflitos.
3. prototipação: os protótipos desenvolvidos na etapa de elicitação de requisitos podem ser aperfeiçoados na etapa de análise, possibilitando uma análise mais completa dos requisitos. Protótipos facilitam o envolvimento dos diferentes stakeholders na elicitação, análise e negociação de requisitos.

Após a análise de requisitos, sendo descobertos conflitos ou problemas, ocorre o processo de negociação de requisitos. Esta atividade visa solucionar problemas advindos do conflito entre os diversos stakeholders, os quais podem atribuir diferentes prioridades aos requisitos. A negociação consiste em que todos os stakeholders entrem em consenso em relação a um conjunto de requisitos definidos, bem como em relação às prioridades definidas para os mesmos. Este trabalho é bastante complexo, pois incide

diretamente em interesses pessoais e obedece, na maioria das vezes, a hierarquias entre os diversos usuários e/ou clientes. Não necessariamente as pessoas que estão no controle da organização são as mais adequadas para definirem prioridades, bem como estabelecerem os requisitos do sistema. O trabalho do engenheiro de requisitos recai neste tipo de dificuldade. Toda e qualquer fonte de informação de requisitos deve ser adequadamente verificada. Os diversos pontos de vista devem ser ponderados para que o documento de requisitos acordados atenda às reais necessidades dos clientes/usuários.

Os processos de elicitação, análise e negociação ocorrem de forma intercalada. Para se chegar a um documento de requisitos que satisfaça aos diversos usuários do sistema, normalmente cada uma das atividades deve ser realizada várias vezes. A meta é estabelecer um consenso sobre os requisitos que estão sendo definidos.

As técnicas existentes na literatura para estas atividades exigem dos engenheiros de requisitos uma formação não somente computacional, mas principalmente humanística, voltada para aspectos de comunicação e necessidades de usuários.

É importante salientar que, para uma melhor realização destas atividades iniciais no processo de Engenharia de Requisitos, são necessárias ferramentas que suportem estas atividades. O grande número de requisitos existentes aponta para a necessidade de utilização de ferramentas CASE, as quais permitem controlar e facilitar o trabalho de engenheiros de requisitos (Kotonya et al. 1997).

2.4.2 Validação de Requisitos

Na elicitação, análise e negociação de requisitos, a preocupação principal residia em que os requisitos certos fossem elicitados e acordados entre clientes e desenvolvedores. Na atividade de validação de requisitos a preocupação é que os requisitos estejam definidos de forma correta. Nesta atividade concentramo-nos na

checagem do documento de requisitos em relação à sua completude, consistência e acurácia.

A atividade de validação de requisitos é a última atividade para certificar-se que o documento final de requisitos satisfaz em todos os aspectos o que o usuário deseja do sistema. Várias técnicas podem ser aplicadas com este fim:

- ***revisões dos requisitos:*** esta é uma das alternativas mais populares para validar requisitos. Consiste na revisão dos requisitos por um grupo de pessoas que analisam, discutem e apontam caminhos para solucionar os problemas encontrados. A partir dessas descobertas, pode-se adotar medidas que solucionem os conflitos. Problemas típicos encontrados nas revisões incluem: incompletude das descrições dos requisitos; ambigüidade; bem como não obediência a padrões da organização.
- ***prototipação***: protótipos são desenvolvidos com o intuito de permitir uma melhor representação dos requisitos de um sistema. Enquanto tradicionalmente, o usuário tem de esperar até etapas no final do processo de desenvolvimento, para visualizar uma versão executável do sistema, com o desenvolvimento de protótipos, usuários podem ter uma idéia antecipada de como o sistema executável funcionará. É observado que a técnica de prototipação geralmente é usada para ajudar nas atividades de elicitação e análise de requisitos. Contudo, ela também pode ser usada para validar os requisitos. Após o documento de requisitos já ter sido definido e acordado, pode-se aperfeiçoar o protótipo desenvolvido na elicitação e análise e utilizá-lo para validar os requisitos, com um auxílio mais efetivo dos usuários/clientes.

- ***testes de requisitos***: comumente, testes de software são realizados no final do processo de desenvolvimento de software. No entanto, recomenda-se desenvolver casos de teste para os requisitos, já no processo de Engenharia de Requisitos. Por exemplo, técnicas baseadas em cenários podem ser usadas para elicitar e analisar requisitos e criar casos de teste para os cenários desenvolvidos. Casos de teste podem ser aplicados de forma simulada nos cenários desenvolvidos. Se surgirem dificuldades para criar os casos de teste, bem como para executá-los através de simulação, há uma grande probabilidade de existirem problemas com os requisitos. É menos oneroso e traumático descobrir estes problemas na atividade de validação do processo de Engenharia de Requisitos, ao invés de em etapas finais do processo de desenvolvimento de software.
- ***validação de modelos:*** geralmente, quando documentamos requisitos, os mesmos são descritos através de linguagem natural e diagramas. Tipicamente, cria-se um documento em linguagem natural descrevendo os requisitos do sistema. Para detalhar melhor o funcionamento do sistema são desenvolvidos modelos do sistema. O desenvolvimento destes modelos está associado à abordagem de desenvolvimento de software adotada. Normalmente, são desenvolvidos modelos de fluxo de dados, modelos de dados semânticos ou modelos inerentes ao paradigma de desenvolvimento, por exemplo, orientação a objetos, métodos formais, etc. Estes modelos necessitam ser validados para demonstrar que os mesmos são consistentes tanto internamente, como externamente. Outrossim, eles devem representar os reais requisitos dos usuários. Na próxima seção, a atividade de documentação de requisitos será melhor descrita.

2.4.3 Documentação e Modelagem de Requisitos

Esta atividade está relacionada a uma descrição detalhada dos requisitos do software, a qual deve ser facilmente entendida por todos os envolvidos. O processo de Engenharia de Requisitos é geralmente guiado por um método adotado para a realização das atividades. Estes métodos possuem uma abordagem sistemática para produzir modelos do sistema. Quando modela-se requisitos, produz-se modelos, os quais podem pertencer a abordagens tais como: modelagem de fluxo de dados, abordagens orientadas a objetos e métodos formais.

A atividade de documentação de requisitos é intercalada em muitos momentos com as atividades de elicitação, análise e negociação de requisitos. O resultado da documentação de requisitos inclui os requisitos acordados representados através de um documento em linguagem natural, bem como uma especificação dos requisitos do sistema. Esta especificação, em geral, consiste de diagramas e modelos pertencentes à metodologia adotada no desenvolvimento. Cada metodologia possui modelos utilizados para documentar diferentes aspectos dos requisitos.

A seguir, apresentamos uma breve descrição dos métodos que podem ser utilizados para documentar/ especificar os requisitos de um sistema:

- ***modelagem de fluxo de dados***: esta é uma das abordagens mais conhecidas na literatura. A abordagem estruturada de desenvolvimento de software foi uma das primeiras abordagens que surgiram com o objetivo de sistematizar a especificação de requisitos, bem como as demais fases do processo de software. O diagrama mais utilizado nesta abordagem é DFD (Diagrama de Fluxo de Dados), o qual objetiva modelar o fluxo de informações que um sistema conterá. É composto por entidades, fluxos de informações, depósitos de dados e processos. Tom DeMarco foi um dos primeiros autores a propor

uma notação gráfica para DFDs (DeMarco 1979). O processo de desenvolvimento de DFDs consiste em inicialmente desenvolver um diagrama de contexto, no qual uma descrição macro do sistema seja obtida e então evoluir a partir desse diagrama, detalhando o fluxo de informações, bem como os processos envolvidos para transformar informações. A idéia básica é chegar em um nível de detalhes, no qual requisitos do sistema possam ser acordados e bem compreendidos. No final de um DFD, deve ser possível evoluir para etapas posteriores do processo de software com todos os requisitos já definidos, ou grande parte deles. Às vezes, um dicionário de dados também acompanha estes DFDs, visando descrever em mais detalhes elementos que compõem cada Diagrama de Fluxo de Dados. A figura 3 mostra uma das notações utilizadas em um DFD.

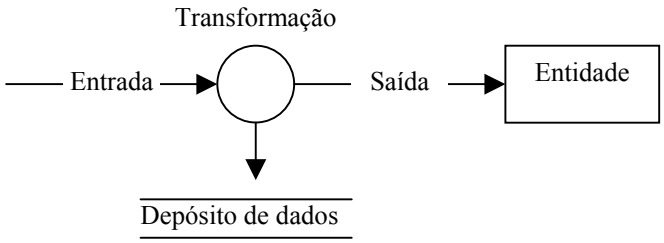

Figura 3. Notações utilizadas em um DFD.

- ***abordagens orientadas a objetos***: uma das abordagens que mais tem crescido, tanto na área acadêmica quanto na industrial, é a abordagem de desenvolvimento de software orientado a objetos. No início da década de 90, diversos métodos propuseram o desenvolvimento através desta abordagem, entre eles a OMT (Rumbaugh et al. 1994), Booch (Booch 1992) e Fusion (Coleman et al. 1994). Basicamente, documentar requisitos através desta abordagem, implica em descobrir os objetos/classes do sistema, bem como

seus relacionamentos. Normalmente, um Diagrama de Classes ou Objetos é criado contendo estes elementos. Cada classe pode conter atributos e operações e os relacionamentos podem incluir mecanismos de herança, composição e outros. Este tipo de estrutura representa um modelo conceitual estático dos requisitos do sistema. Dependendo do método orientado a objetos utilizado, diferentes modelos e diagramas são utilizados para documentar artefatos originados em outras etapas do processo de desenvolvimento, tais como: projeto e implementação.

Para tornar as notações orientadas a objetos mais padronizadas e possibilitar uma unificação das notações dos vários métodos existentes, foi definida pelo Grupo de Gerenciamento de Objetos (OMG), uma linguagem de modelagem de objetos denominada de UML (Booch et al. 1999). Esta linguagem surgiu da unificação dos métodos dos autores Ivar Jacobson, Grady Booch e James Rumbaugh. A Linguagem de Modelagem Unificada – UML apresenta uma série de notações para o desenvolvimento orientado a objetos. Atualmente, há uma tendência em que a documentação/modelagem dos requisitos através da abordagem orientada a objetos, na sua maioria utilize a linguagem UML.

- ***técnicas formais***: as abordagens tradicionais de modelagem de requisitos fundamentam-se em diagramas, bem como em descrições textuais, as quais documentam os requisitos. Outra abordagem para documentar requisitos de forma precisa é descrever as funções e o comportamento do sistema, utilizando-se de uma sintaxe e uma semântica formal matemática (Clarke et al. 1996). Várias linguagens utilizando estes princípios são utilizadas para especificar sistemas de software, entre as quais podemos citar Z (Spivey et al. 1992) (Diller 1994) e VDM (Jones 1990). O objetivo é

especificar/documentar os requisitos de forma que o rigor matemático propicie diminuir as ambigüidades e incompletudes de um documento de requisitos de usuário.

Através de uma linguagem de especificação formal, os requisitos são documentados utilizando uma sintaxe, semântica e um mecanismo de relações que permitem um maior grau de precisão dos requisitos e, conseqüentemente do sistema que será desenvolvido.

2.4.4 Gerenciamento de Requisitos

Esta atividade é uma das mais importantes do processo de Engenharia de Requisitos. Tem como meta principal o controle e gerenciamento de mudanças nos requisitos (Toranzo 2002).

Um dos maiores problemas atuais no desenvolvimento de software reside no fato de que os requisitos de software são modificados, seja por questões de melhoria das funções do sistema, por solicitações do usuário ou devido a correções de erros encontrados. Nestes casos, mudanças podem gerar efeitos colaterais, bem como propagar-se negativamente para outras partes do software. O controle dos requisitos descobertos e analisados no processo de Engenharia de Requisitos é fundamental para que as mudanças sejam adequadamente tratadas e seus impactos corretamente avaliados. Isto implica, necessariamente, em poder rastrear os requisitos ao longo de todos os artefatos produzidos no processo de Engenharia de software. Este rastreamento de requisitos deve ocorrer em ambos os sentidos, dos requisitos iniciais para os artefatos desenvolvidos, bem como dos artefatos para as fontes que originaram os requisitos.

Gerenciar requisitos significa poder escrever e acompanhar um requisito ao longo de todo o processo de desenvolvimento, enquanto existir. Isto garante

documentos de requisitos mais confiáveis e gerenciáveis, aspectos importantes para a obtenção de produtos de software de qualidade.

De forma resumida, podemos descrever os objetivos do gerenciamento de requisitos como:

- gerenciar mudanças para os requisitos acordados;
- gerenciar o relacionamento entre requisitos;
- gerenciar as dependências entre o documento de requisitos e os demais documentos produzidos no processo de Engenharia de Requisitos.

No processo de gerenciamento de requisitos é fundamental que cada requisito possua algum tipo de identificação única (Kotonya et al. 1997). Esta identificação é o meio adotado para poder rastrear, bem como avaliar os impactos advindos de mudanças. Para grandes sistemas, nos quais o número de requisitos a serem gerenciados é muito grande, é necessário que os mesmos sejam armazenados em uma base de dados e sejam registradas as ligações entre os requisitos relacionados. Atualmente, verifica-se que na grande maioria dos sistemas, faz-se necessário o uso de ferramentas CASE, que apóiem o processo de gerenciamento de requisitos. Grandes quantidades de requisitos não são gerenciáveis sem a utilização de alguma ferramenta computacional de apoio. Como exemplo de ferramentas existentes atualmente no mercado usadas com este fim, podemos citar Requisite Pro (Rational 1999), Doors (Quality 1999), QSSRequireit (http://www.qssrequireit.com) e Caliber-RM (www.tbi.com).

É bastante aceito também que as ferramentas CASE, para gerenciar requisitos, devem evoluir e as mesmas devem ser integradas com outras ferramentas de apoio às demais etapas do processo de Engenharia de software. O aspecto mais relevante é que documentos produzidos/mantidos por uma ferramenta de gerenciamento de requisitos,

devem estar relacionados, de alguma forma, com outros artefatos do processo de software. Este é ainda um desafio nesta área.

É importante salientar que a gerência de requisitos é uma atividade considerada essencial na Engenharia de software como um todo. Mostra disto, é a inclusão desta atividade nos principais mecanismos de avaliação de qualidade de produtos e processos de softwares, tais como: CMM (Humphrey 1989), ISO (ISO/IEC 1991), (NBR ISO/IEC 1995) (Oskarsson et al. 1995) e SPICE (ISO/IEC 1999). No entanto, estes padrões de qualidade ainda não definem em sua plenitude todas as características essenciais que devem ser consideradas quando gerenciamos requisitos. Por outro lado, há uma dificuldade muito em grande em estabelecer políticas de gerenciamento de requisitos. Isto é conseqüência da grande quantidade de informações tratadas, bem como da própria complexidade associada à execução das atividades de gerenciamento de requisitos.

2.5 Preocupações atuais na Engenharia de Requisitos

Nos últimos anos, tem ficado evidente a necessidade de aprofundar de forma mais sistemática os estudos relacionados com as atividades de Engenharia de Requisitos. Um dos principais motivos é o aumento crescente dos custos advindos da correção de problemas associados com requisitos inadequadamente elicitados, analisados e especificados. O que se vê são usuários insatisfeitos com softwares que não se adequam às suas reais necessidades (Kozlenkov et al. 2002).

Aspectos relacionados ao ambiente organizacional, no qual o software está inserido, são comumente desconsiderados ou avaliados de forma incompleta. É extremamente importante estabelecer uma correspondência entre o ambiente organizacional e o sistema de software que esteja sendo desenvolvido para executar neste ambiente. Isto implica em avaliar objetivos e metas organizacionais e apontar de

que forma sistemas de software podem satisfazer estes objetivos. Fazer esta ligação não é um processo trivial, pois envolve lidar com diferentes stakeholders tais como usuários, clientes e engenheiros de requisitos, os quais podem possuir diferentes níveis de formação e interesses.

Podemos também verificar que a maioria dos métodos e processos de desenvolvimento de software (D'Souza 1998) (Jacobson et al. 1999) (Kruchten 2000) existentes não tratam de forma satisfatória estas questões. Em alguns casos, são fornecidas técnicas para modelar aspectos organizacionais e de negócio, mas não se estabelece uma clara conexão entre estes elementos e a especificação dos requisitos do sistema. Pesquisas nesta área são importantes e têm sido incentivadas pela comunidade de Engenharia de Requisitos.

Outro ponto relevante, diz respeito ao aprimoramento e desenvolvimento de recursos técnicos e humanos mais apropriados na elicitação, documentação e validação de requisitos. Isto implica, não somente, em técnicas mais efetivas aplicadas em atividades no processo de Engenharia de Requisitos (Goguen et al. 1993) (Kotonya et al. 1997) (Sommerville et al. 1997), mas também em uma melhor formação sociológica e multidisciplinar dos profissionais envolvidos. Umas das grandes dificuldades, ainda reside no fato de que engenheiros de requisitos têm dificuldade em elicitar requisitos de clientes e usuários. Isto ocorre por questões sociais e organizacionais, as quais devem ser adequadamente tratadas para possibilitar o desenvolvimento de sistemas que satisfaçam realmente as metas organizacionais relevantes para a organização.

Neste aspecto, algumas pesquisas têm apontado técnicas como a etnografia (Sommerville et al. 1993) (Sommerville et al. 1999) (Hughes et al. 1995) para auxiliar engenheiros de requisitos. Este tipo de técnica tem como base a observação, descrição e análise detalhada das práticas de trabalho em uma organização. Objetiva-se propiciar a

engenheiros de requisitos entender e documentar a *linguagem* dos usuários do sistema e sua relação com as tarefas do trabalho das pessoas. Assim, familiarizado com o ambiente e linguagem dos usuários, o engenheiro de requisitos pode desenvolver documentos de requisitos com menos ambigüidades, incompreensões e incompletudes. Técnicas com esse fim vêm sendo pesquisadas, mas aponta-se como uma das principais dificuldades para a sua utilização; o tempo consumido para aplicação das mesmas. Normalmente, engenheiros de requisitos possuem prazos de entrega como elemento de pressão e limitante na realização do seu trabalho.

Existe também, atualmente, uma preocupação em relação ao nível de precisão adequado que deve ser utilizado em atividades na Engenharia de Requisitos. Quando requisitos são elicitados, documentados e validados, podemos optar pelo uso de técnicas com maior ou menor precisão ou podemos até mesmo utilizar métodos formais, os quais possuem sintaxe e semântica precisas e passíveis de validação. Por outro lado, técnicas tradicionais são mais informais e os artefatos produzidos por estas técnicas são difíceis de serem verificados e validados com completa segurança e exatidão. Estabelecer o nível certo de formalidade no desenvolvimento de para cada tipo de software, bem como possibilitar uma integração e/ou utilização conjunta de métodos formais e não formais é uma aspecto bastante crítico.

2.6 Considerações Finais

Neste capítulo apresentamos os principais conceitos associados com a Engenharia de Requisitos. A Engenharia de Requisitos (ER) é um campo de engenharia relativamente novo quando comparado a outras disciplinas de engenharia já consolidadas tais como Engenharia Civil e Engenharia Elétrica. Por isso, as pesquisas são crescentes, buscando encontrar processos, técnicas e ferramentas que possam

facilitar o trabalho de engenheiros de requisitos na definição de documentos de requisitos mais consistentes. Neste sentido, técnicas de modelagem organizacional tais como i* (Yu 1995) (Castro et al. 2000) (Mylopoulos et al. 2000), têm sido desenvolvidas para auxiliar os diversos *stakeholders*, a entender as intenções, razões e relacionamentos no ambiente organizacional. Acredita-se que desta forma, engenheiros de requisitos possam elicitar, analisar e especificar requisitos de sistemas, compreendendo melhor o ambiente organizacional e seus relacionamentos, bem como considerando requisitos e objetivos organizacionais previamente estabelecidos.

Consideramos, contudo, que requisitos organizacionais precisam também ser relacionados de uma forma mais sistemática a requisitos funcionais de sistemas de software. É neste sentido que propomos melhorar as atividades de *elicitação, análise e documentação de requisitos*, do processo de Engenharia de Requisitos, através da integração de modelos organizacionais com técnicas baseadas em cenários. Esta proposta em detalhes estará descrita no capítulo 5.

No próximo capítulo, apresentamos um relato de algumas técnicas baseadas em cenários integradas com abordagens orientadas a objetivos, as quais podem auxiliar engenheiros de requisitos na realização de suas atividades. Apresentamos as principais vantagens e limitações de algumas destas técnicas, com um enfoque especial à descrição da técnica de Caso de Uso, a qual será integrada na nossa proposta com modelos organizacionais desenvolvidos através de i*.

Salientamos também que a motivação para trabalhos na área de Engenharia de Requisitos tem crescido ao longo dos últimos anos. Isto se deve, principalmente, ao fato de que grande parte dos problemas atribuídos a sistemas que não satisfazem usuários adequadamente são originados nas atividades realizadas nesta fase.

Capítulo 3

Estudo de Técnicas Baseadas em Cenários Integradas com Abordagens Orientadas a Objetivos

Técnicas baseadas em cenários têm sido utilizadas na Engenharia de Requisitos para entender, modelar e validar os requisitos de usuários. Algumas abordagens propondo a utilização de cenários para elicitação e validação de requisitos incluem (Haumer et al. 1998) (Jacobson 1995) (Leite et al. 1997) (Breitman et al. 1998) (Rolland et al. 1998) (Sutcliffe et al. 2002).

Objetiva-se com o uso de cenários descrever as ações em um ambiente relacionadas a um sistema atual ou a um sistema a ser desenvolvido. Normalmente, a linguagem utilizada para estas descrições é a linguagem natural. No entanto, técnicas baseadas em cenários, tais como: Caso de Uso (Jacobson 1995) e o projeto CREWS (*Cooperative Requirements Engineering With Scenarios*) (Ralyté 1999) também utilizam diagramas e notações gráficas.

Os trabalhos na Engenharia de Requisitos relacionados com técnicas baseadas em cenários têm sido crescentes. Desde os anos 80, os resultados destes trabalhos vêm sendo divulgados em jornais e revistas da área. Há alguns anos, Caso de Uso, o qual é um tipo de técnica que utiliza cenários, tem sido adotada como integrante do modelo padrão de linguagem de modelagem para a orientação a objetos: UML. Esta é uma das provas mais fortes da relevância de técnicas baseadas em cenários. Assim, cada vez mais, reconhece-se em cenários, uma forma efetiva de auxílio no entendimento e especificação de requisitos de usuários.

Contudo, mais recentemente, cenários vem sendo utilizados em conjunto com abordagens de análise de requisitos orientadas a objetivos (Anton 1997), (Potts 1999) (Cockburn 2000) (Kaiva et al. 2002). Abordagens orientadas a objetivos têm como foco expressar os "porquês" associados com a construção de sistemas (Yu et al. 1998). Estes "porquês" expressam as razões e justificativas para sistemas propostos. Objetivos sob este ponto de vista são metas que clientes e usuários possuem em relação a sistemas de software e que podem ser descritos e refinados em cenários. A idéia fundamental é que, focalizando-se nos objetivos no início do processo de Engenharia de Requisitos, possibilita-se que analistas comuniquem-se com os stakeholders utilizando uma linguagem baseada nos conceitos (objetivos), com os quais ambos estão familiarizados e sentem-se confortáveis. Assim, com base nos objetivos descobertos para sistemas, pode-se derivar e descrever cenários de uso destes sistemas; e ainda, analisando-se estes cenários, pode-se descobrir novos objetivos para sistemas computacionais. Este processo é iterativo na busca de novos requisitos de sistemas.

Neste capítulo, descrevemos algumas propostas que integram técnicas baseadas em cenários e abordagens orientadas a objetivos. Objetivamos criar uma base de entendimento para a nossa proposta, bem como apresentar as abordagens que auxiliaram-nos na definição das heurísticas para integração de modelos organizacionais e casos de uso e seus cenários. Inicialmente, na seção 3.1, apresentamos a técnica de caso de uso enfatizando alguns aspectos de sua escrita, conforme proposto por Cockburn (2000). A principal contribuição de Cockburn é a associação de um objetivo específico ao caso de uso a ser descrito. Objetivos de casos de uso nessa abordagem também são classificados em níveis, evitando que usuários se percam no que exatamente querem obter. Esta seção é particularmente importante, pois nesta tese propomos integrar modelos organizacionais em i* (descritos no capítulo 4) com casos

de uso. Na seção 3.2, apresentamos a abordagem do Projeto CREWS (*Cooperative Requirements Engineering With Scenarios*) (Ralyté 1999) (Ralyté et al. 1999), denominada *L' Ecritoire*, a qual adota um acoplamento bidirecional para facilitar a navegação entre objetivos e seus cenários associados. Na seção 3.3 apresentamos a técnica GBRAM (Goal-Based Requirements Analysis Method) (Anton 1997), a qual usa uma topografia de objetivos para estruturar e organizar requisitos na forma de cenários, obstáculos e restrições de objetivos. A tipografia apresentada permite que analistas encontrem e ordenem objetivos, bem como utilizem cenários para auxiliar na descoberta de novos objetivos e elaboração de requisitos (Anton 2001). Na seção 3.4, apresentamos a proposta de cenários em Leite et al. (1997) (Breitman et al. 1998), a qual define cenários como ferramenta para definir "*baselines*" de requisitos. Apresenta-se nesta abordagem um *template* de descrição de cenários, o qual inclui entre outros aspectos, o objetivo a ser obtido com o cenário. Na seção 3.5, apresentamos as considerações finais do capítulo.

3.1 Casos de Uso

Casos de Uso em UML (Booch et al. 1999), são utilizados para descrever o uso de um sistema por atores. Um **ator** representa qualquer elemento externo que interage com o sistema. Um **caso de uso** descreve uma seqüência de passos/operações que um usuário realiza quando interage com um sistema, visando realizar uma determinada tarefa ou alcançar um objetivo. Dessa forma, o aspecto comportamental de um sistema a ser desenvolvido pode ser descrito através de casos de uso. No entanto, estas descrições não tratam da questão de como as interações entre o usuário e o sistema serão implementadas. Fases posteriores à etapa de Engenharia de Requisitos, tais como: Projeto e Implementação, focalizarão neste aspecto.

Um caso de uso pode gerar vários cenários. Cenários estão para casos de uso, assim como instâncias estão para classes, significando que um cenário é basicamente uma instância de um caso de uso. Um caso de uso envolve uma situação de utilização do sistema por um ator. Nesta situação, vários caminhos podem ser seguidos, dependendo do contexto na execução do sistema. Estes caminhos são os possíveis cenários do caso de uso. Considera-se que o caminho básico para realizar um caso de uso, sem problemas e sem erros em nenhum dos passos da seqüência, é denominado de **cenário primário**. Neste tipo de cenário, a execução dos passos para realizar a funcionalidade básica do caso de uso, é obtida com sucesso. Por outro lado, caminhos alternativos, bem como situações de erro, podem ser representados através de **cenários secundários**. Cenários secundários descrevem seqüências alternativas e de erros, que podem ocorrer em um cenário primário associado com um caso de uso. Cenários secundários podem ser descritos separadamente ou como extensão da descrição de um cenário primário. Se um cenário secundário é bastante complexo e inclui um conjunto considerável de passos, é conveniente descrevê-lo separadamente.

Outras técnicas também podem ser usadas na Linguagem de Modelagem Unificada (UML), para refinar fluxos de eventos em Casos de Uso. A idéia consiste, basicamente, em incluir relacionamentos que permitam descrever diversos aspectos de comportamento entre casos de uso. Os relacionamentos apontados em UML incluem:

- relacionamento do tipo **<<*include*>>:** quando for detectado no sistema um conjunto de **passos comuns** a vários casos de uso, pode-se criar um caso de uso com estes passos, com potencial para ser reutilizado por outros casos de uso. A idéia consiste em encapsular em um **caso de uso específico**, um comportamento comum a vários casos de uso, estabelecendo que os demais casos de uso do sistema podem fazer uso do mesmo (i.e. incluí-lo), quando necessário.

- relacionamento do tipo (<<***extend***>>): utilizamos este tipo de relacionamento quando existe uma **seqüência opcional ou condicional** de passos que queremos incluir em um caso de uso. Esta seqüência de passos deve ser descrita em um caso de uso específico, que poderá ser utilizado por outros casos de uso em certo ponto de sua execução. O uso do caso de uso extendido ocorre devido a uma situação de comportamento opcional ou condicional.
- relacionamento do tipo **<<*generalization*>>:** generalização entre casos de uso tem o mesmo significado de generalização entre classes na orientação a objetos. Isto significa que um caso de uso "filho" **herda** o comportamento e estrutura do caso de uso "pai". Considera-se que um caso de uso "filho" é uma **especialização** do caso de uso "pai", podendo adicionar nova estrutura e comportamento, bem como modificar o comportamento do caso de uso "pai".

Da mesma forma que permite-se o uso do mecanismo de generalização entre casos de uso, pode-se usar o relacionamento de generalização entre **atores** representados em diagramas de casos de uso. A figura 4 apresenta as notações básicas utilizadas para descrever Casos de Uso.

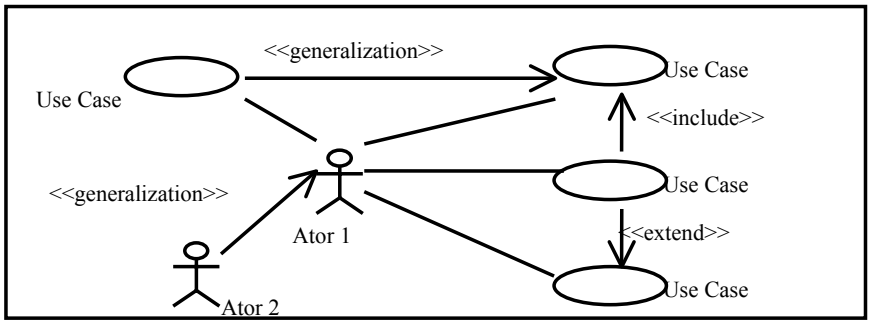

Figura 4. Notações para Casos de Uso em UML .

Na figura acima, o ator 2, por exemplo, pode ser uma especialização do ator 1. Neste caso, o ator 2 herda toda a estrutura e comportamento do ator 1 e pode adicionar nova estrutura e comportamento em relação ao ator 1. Outras técnicas adicionais

propostas em UML para representar e descrever melhor casos de uso podem ser vistas em Booch et al. (1999).

Um possível processo de construção de Casos de Uso inicia com a descoberta dos atores do sistema e prossegue com a descoberta dos Casos de Uso associados com estes atores. Para cada ator, são encontrados todos os Casos de Uso relacionados ao mesmo. Isso ocorre porque cada ator requer do sistema algumas funcionalidades, sendo que os passos necessários para obter estas funcionalidades são descritos através Casos de Uso.

O segundo passo consiste em definir os caminhos básicos (cenários primários) e, posteriormente, os caminhos alternativos (cenários secundários), para cada um dos Casos de Uso.

O terceiro passo envolve revisar descrições de aspectos comportamentais de casos de uso encontrando relacionamentos do tipo <<*include*>>, <<*extend*>> e <<*generalization*>>. Este processo é, geralmente, realizado adotando-se o princípio de desenvolvimento de software iterativo e incremental. Depois de definidos todos os Casos de Uso e atores do sistema, desenvolve-se um modelo de Casos de Uso utilizando as notações apresentadas na figura 4.

No entanto, a tarefa de descobrir e descrever casos de uso não é tão simples (Fantechi et al. 2002) (Woo et al. 2002), pois na maioria das situações, exige um certo grau de experiência de engenheiros de requisitos. O primeiro aspecto é descobrir quais são os atores que realmente desejam obter algum serviço em relação ao sistema. Estes atores, também chamados de atores primários, são os mais importantes, pois têm um objetivo claro em relação ao sistema e interagem com o sistema para obter este objetivo. Alguns atores, no entanto, podem desempenhar um papel de suporte de algum serviço em relação ao sistema ou podem ter interesses no comportamento do sistema. Como

exemplo, poderíamos citar atores tais como: um software que fornece serviço na WEB, uma impressora de alta velocidade, o departamento da empresa que tem interesse no sistema, etc. Estes atores colaboram na realização de casos de uso para atores primários. Neste ponto, é importante diferenciar estes tipos de atores e concentrar os esforços na descrição de atores primários e na descoberta de casos de uso para os mesmos. Para efeitos de simplificação, chamaremos de agora em diante atores primários de apenas atores e quando for outro tipo de ator, deixaremos isto explícito.

Outra dificuldade é definir quais são os casos de uso que realmente satisfazem os interesses dos atores, proporcionando-lhes algo relevante como resultado da execução do caso de uso. Uma das alternativas que pode ajudar nesta tarefa é apontada em Cockburn (2000). Parte-se do princípio que *stakeholders*, e mais especificamente usuários e clientes, possuem **objetivos** em relação ao sistema a ser desenvolvido. Estes objetivos podem ser de mais alto nível ou de baixo nível, e são estes objetivos os que originam casos de uso para os atores interessados.

Segundo Cockburn (2000), o nível mais importante do ponto de vista de definição de requisitos de sistemas computacionais é o ***objetivo de usuário***, o qual representa o que o ator do sistema está querendo obter de relevante e de valor (funcionalidade), no uso do sistema computacional. Este nível corresponde ao que poderíamos chamar de "tarefa do usuário" ou "processo de negócio elementar" (em um caso de uso de negócio). Um objetivo de usuário está relacionado com a questão: "O usuário poderá ir embora satisfeito depois de ter obtido este objetivo?". Outra pergunta que pode auxiliar na identificação de objetivos de usuários é: "A obtenção do objetivo satisfaz as responsabilidades do usuário no seu trabalho? ".

> *Tipicamente, efetuar "log on" no sistema não é um objetivo de usuário. Por exemplo, "logar no sistema" 40 vezes normalmente não satisfaz as*

responsabilidades que um usuário possui no seu trabalho nem o propósito do mesmo no uso do sistema. No entanto,"registrar um novo cliente" é um objetivo de usuário significante. Registrar 40 novos clientes soa bastante importante e significativo para usuários no uso do sistema. Outro exemplo de objetivo de usuário também poderia ser "o cliente deseja fazer um pedido", em uma aplicação de comercio eletrônico (ver exemplo no capítulo 6).

Um ***objetivo de contexto*** agrega uma série de objetivos de usuários associados a um mesmo objetivo de mais alto nível. Este tipo de objetivo é bastante útil para prover um controle de um conjunto de objetivos de usuários. Objetivos de contexto mostram o contexto no qual objetivos de usuários irão operar, bem como a seqüência de objetivos relacionados. Tipicamente casos de uso com objetivos de contexto, executam durante horas, dias, semanas, meses ou anos.

Um exemplo deste tipo de objetivo poderia ser "Tratar um Pedido de Indenização" em um sistema de seguro de veículos. Este objetivo englobaria uma série de objetivos de usuários relacionados tais como: encontrar apólice de seguro, capturar dados da perda, atualizar dados do pedido de indenização, atualizar valores do pagamento de indenização, etc. Outro exemplo deste tipo de objetivo pode ser também "Processar pedidos realizados pela internet" em uma aplicação de comércio eletrônico (ver exemplo no capítulo 6).

Finalmente, ***objetivos de subfunção*** são aqueles requeridos para levar a cabo *objetivos de usuários*. Uma subfunção é um sub-objetivo ou passo em um cenário, abaixo do nível principal de interesse de usuários. De forma geral, este tipo de objetivo de caso de uso é normalmente utilizado por muitos outros casos de uso, em nível de objetivo de usuário.

Exemplos destes objetivos são "logar no sistema", "encontrar um produto através de uma pesquisa", "encontrar um cliente através de uma pesquisa" ou "salvar como arquivo". Em uma aplicação de comércio eletrônico (ver exemplo no capítulo 6), um exemplo deste tipo de objetivo pode ser: "O cliente deseja pesquisar (via navegador) o catálogo".

O *template* de especificação de caso de uso proposto por Cockburn (2000) e descrito na figura 5, define explicitamente objetivos de casos de uso, bem como os níveis associados com estes objetivos. As demais informações presentes no *template* também são importantes para tornar a descrição de casos de uso o mais clara possível. Salientamos que adotaremos este *template* como modelo para a especificação de casos de uso na nossa proposta de derivação de casos de uso a partir de modelos organizacionais. Este *template* foi escolhido, principalmente, por se enquadrar na nossa visão associada à necessidade de definir explicitamente um objetivo associado com cada caso de uso. O objetivo de cada caso de uso é explicitamente definido no campo "objetivo no contexto" no *template*. Consideramos também que a atribuição de níveis de abstração (campo "nível" no *template*) para cada objetivo de caso de uso no template proposto por Cockburn (2000), é bastante útil, permitindo expressar e gerenciar mais adequadamente o nível de abstração associado com cada caso de uso.

Contudo, estabelecer o nível adequado do objetivo para cada caso de uso, é um trabalho que exige experiência e um alto grau de interação em relação aos interesses de *stakeholders*. Uma técnica que pode ajudar nesta tarefa é construir uma lista ator-objetivo associada com o sistema em desenvolvimento. Pode-se também criar uma tabela com os elementos desta lista, acrescentando-se colunas para incluir: a prioridade do objetivo no contexto do desenvolvimento; a necessidade de obter o objetivo do ponto

de vista da organização e do negócio; bem como a dificuldade técnica que pode estar associada na construção do caso de uso correspondente.

***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 5. Template de caso de uso (Cockburn 2000).

No entanto, a maior dificuldade está relacionada a como e de que forma os objetivos em relação a um sistema pretendido podem inicialmente ser descobertos. Tradicionalmente, cabe a engenheiros de requisitos, com base principalmente em suas experiências, descobrir estes objetivos, para poder assim então descrever os casos de uso para o sistema. Neste sentido, acreditamos que uma alternativa viável é iniciar o processo de descoberta de casos de uso com base nos objetivos e demais elementos representados em modelos organizacionais. O uso das informações definidas em modelos organizacionais pode facilitar a descoberta e descrição de casos de uso, bem como tornar este processo mais sistemático. Salientamos que, as heurísticas apresentadas na literatura (Jacobson et al. 1999) (Schneider et al. 1998) para desenvolver casos de uso, em geral ainda tornam o processo ad-hoc. Na maioria das situações, casos de uso são desenvolvidos sem considerar os requisitos organizacionais previamente definidos por uma organização. Tendo em vista estas dificuldades e carências associadas com a técnica de caso de uso, concentramo-nos nesta tese em desenvolver casos de uso a partir da observação e análise dos objetivos e demais elementos definidos em modelos organizacionais. Uma descrição detalhada da nossa proposta será apresentada no capítulo 5.

Um aspecto também importante no processo de desenvolvimento de casos de uso é situar esta técnica no contexto da Engenharia de Requisitos. Casos de Uso podem ser uma parte do documento de requisitos que deve ser desenvolvido no processo de Engenharia de Requisitos. Representam basicamente aspectos funcionais e comportamentais do sistema a ser desenvolvido. Em Cockburn (2000), apresenta-se uma representação gráfica para os elementos que compõem o documento de requisitos, e de que forma casos de uso fazem parte desse documento. Fizemos algumas adaptações

que consideramos adequadas, visando representar melhor o propósito do nosso trabalho. Esta visão é representada na figura 6.

O modelo da figura 6 representa uma "roda," na qual o eixo central corresponde aos casos de uso desenvolvidos para um sistema de software. Casos de uso, basicamente representam requisitos funcionais, mas estão conectados e relacionados a outros requisitos, tais como: requisitos organizacionais, requisitos não-funcionais e requisitos de formato de dados. Casos de Uso também podem auxiliar na definição de requisitos para interface de usuários (UI), bem como na definição de formatos e protocolos de I/O. A utilização de casos de uso no processo de Engenharia de Requisitos pode também ajudar em outras atividades realizadas em etapas posteriores no processo de desenvolvimento de software. Como exemplo, podemos destacar o desenvolvimento de Projetos de Casos de Teste e Projetos de Interface de Usuário.

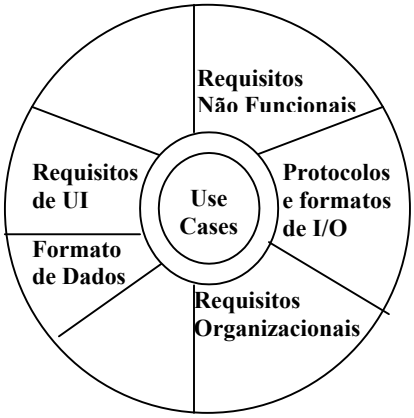

Figura 6. Relacionando Casos de Uso com outros tipos de Requisitos.

É consensual, que casos de uso não são suficientes para detalhar todos os elementos que devem ser definidos no processo de Engenharia de Requisitos. No entanto, as vantagens do uso desta técnica, como também de outras técnicas baseadas em cenários, é que podemos juntamente com as descrições de interações entre um usuário e o sistema, relacionar outros tipos de requisitos, tais como: requisitos não

funcionais e organizacionais, bem como evoluir posteriormente para outros artefatos no processo de desenvolvimento. Como exemplo, podemos citar o Processo Unificado (*Unified Process*) (Jacobson et al. 1999), o qual adota a descrição de casos de uso como fonte de informações para gerar elementos, entre os quais podemos citar: diagramas de classes, diagramas de seqüência bem como descrições arquiteturais do software.

3.2 Proposta de Cenários apresentada no projeto CREWS (Cooperative Requirements Engineering With Scenarios)

O projeto CREWS (Cooperative Requirements Engineering With Scenarios) foi desenvolvido na Europa tendo como objetivo desenvolver, avaliar e demonstrar a aplicabilidade de métodos e ferramentas para a elicitação e validação de requisitos de software, através de técnicas baseadas em cenários. Os resultados oriundos do projeto apontaram soluções para os problemas existentes no desenvolvimento e uso de cenários. Parte-se do princípio que, a utilização de cenários pode facilitar a integração mais eficiente dos diferentes Stakeholders, bem como possibilitar o desenvolvimento de documentos de requisitos mais completos.

Os trabalhos em CREWS foram motivados pela dificuldade de elaborar e integrar cenários, de forma que as vantagens do uso da técnica fossem claramente observadas e acordadas. Desta forma surgiram quatro abordagens focando cenários.

Duas destas abordagens apontam técnicas que permitem elicitar requisitos a partir de cenas do mundo real (Haumer et al. 1998), bem como através da descrição de cenários em linguagem natural (Rolland et al. 1998). As outras duas abordagens lidam com a validação de requisitos através da geração sistemática de cenários, acompanhadas por *walkthrough* de requisitos (Suftcliffe 1998) e animação de cenários (Dubois 1998). A base do método CREWS, integrando as abordagens propostas no projeto, é apresentada em Ralyté (1999).

A contribuição mais efetiva e chave na proposta do Projeto CREWS é a abordagem denominada L'Ecritoire. Esta abordagem tem como meta permitir a elicitação de requisitos, através de uma estratégia centrada no uso integrado de modelagem de objetivos e cenários (Rolland et al 1998). Na Engenharia de Requisitos, tanto a modelagem de objetivos (Potts 1997) quanto técnicas baseadas em cenários, têm sido utilizadas para elicitar requisitos. Apesar de serem técnicas com características e enfoques diferentes, ambas objetivam elicitar requisitos a partir de uma análise do contexto, no qual o sistema operará. Considerando este aspecto, L'Ecritoire apresenta uma proposta para o uso conjunto destas técnicas na elicitação de requisitos.

Propõe-se em L'Ecritoire a descoberta de requisitos usando o acoplamento bidirecional Objetivo-Cenário, permitindo-se evoluir de objetivos para cenários e vice versa. Requisitos são descobertos e descritos em duas etapas: quando um objetivo é descoberto, um cenário é associado com o mesmo, e quando a descrição do cenário é concluída e acordada (autorizada), o mesmo é analisado para descoberta de novos objetivos envolvidos. A descoberta de objetivos e a descrição e análise de cenários são passos complementares. O processo é repetido à medida que novos objetivos são descobertos através do ciclo, que inclui descoberta_de_objetivos e autorização_de_cenários. Este processo é conduzido por uma série de diretrizes definidas na abordagem. Os principais elementos da abordagem L'Ecritoire são apresentados a seguir.

O conceito de *objetivo* é definido como "algo que um *stakeholder* espera obter/alcançar no futuro". Um *cenário* é definido como "um possível comportamento associado a um conjunto de interações entre vários agentes" (Rolland 1998). É composto de uma ou várias ações, sendo uma ação uma interação de um agente com outro. A combinação de ações em um cenário descreve um único caminho. Um cenário

é caracterizado por um estado inicial e um estado final. O estado inicial define uma precondição necessária à execução do cenário e o estado final define o estado obtido no final da execução de um cenário. Distingue-se em L'Ecritoire, cenários normais e cenários excepcionais. Um cenário normal define um caminho para obter um objetivo em um situação normal e de sucesso. Cenários excepcionais definem caminhos nos quais algo excepcional ocorre e impede a obtenção do objetivo com sucesso.

O principal componente da abordagem é denominado de *Chunk de Requisito (RC)*[3]. Um *chunk de requisito* representa uma parte ou fragmento da definição de requisitos de um sistema. Podemos visualizá-lo como um par <G,Sc> onde G é um objetivo e Sc é um cenário. Como um objetivo é considerado algo intencional e um cenário como uma descrição operacional, um RC é um possível caminho (cenário) para a obtenção de um objetivo. O RC é a forma encontrada para representar o acoplamento bidirecional entre objetivo-cenário, comentado anteriormente. Na figura 8, apresentamos um exemplo de Chunk de Requisito, no contexto de um sistema pretendido para uma máquina de transações automáticas bancárias (ATM), bastante conhecido na literatura.

Outro aspecto importante que a abordagem propõe, é a diferenciação de *níveis de abstração e classificação* dos *chunks de Requisito.* São introduzidos três níveis de abstração denominados de *Contextual*, *Funcional* e *Físico*. O nível contextual identifica serviços que o sistema a ser desenvolvido deveria prover para a organização, bem como também, as razões associadas com a obtenção destes serviços/objetivos. Neste nível, a principal preocupação é explorar as possíveis alternativas de projeto que podem satisfazer os objetivos de negócio ou objetivos organizacionais previamente definidos.

[3] Requirement Chunk

O refinamento de RCs neste nível pode derivar RCs no nível de abstração mais baixo (funcional).

O nível funcional tem como foco as interações entre os usuários e o sistema. Um *RC* neste nível contém o par (G,Sc), que representa um objetivo de usuário (G) e o cenário (Sc), descrevendo o fluxo de interações entre o usuário e o sistema para alcançar o objetivo. Captura-se em um *RC* funcional, um caminho possível para satisfazer objetivos de usuários.

No nível físico, o foco é a representação das ações internas do sistema necessárias para realizar interações modeladas em *RCs* no nível funcional. Detalha-se, neste nível, um possível caminho, que o sistema internamente pode adotar para realizar uma determinada interação, contida em um cenário definido em um *RC* no nível (funcional) anterior.

Assim, os requisitos são elicitados nos três níveis de abstração, representando um tipo de Chunk de Requisito.

Toda abordagem efetiva de elicitação de requisitos precisa propor meios para organizar e relacionar o conjunto de requisitos. Para esse fim, a abordagem L'Ecritoire define três tipos de *relacionamentos entre RCs* denominados de: *composição, alternativa e refinamento*. Estes relacionamentos, na verdade, são também estratégias de descoberta de objetivos, possibilitando assim a execução do ciclo descoberta_de_objetivo/autorização_de_cenário, incrementado a hierarquia de *RCs* definidos para o sistema. Os relacionamentos do tipo composição e alternativa, representam respectivamente estruturas AND e OR, entre *RCs*. São relacionamentos que estruturam de forma horizontal os *RCs*. O relacionamento AND, relaciona *RCs,* que dependem uns dos outros e que juntos definem completamente uma funcionalidade do sistema. Tipicamente, *RCs* originados de refinamentos em um cenário funcional, são

relacionados via relacionamento AND. *RCs* relacionados através de OR representam caminhos alternativos para satisfazer um mesmo objetivo.

O relacionamento do tipo *refinamento* entre *RCs*, determina diferentes níveis de abstração para cada *RC*. Este tipo de relacionamento estabelece uma estrutura vertical entre os *RCs*. Um *RC*, por exemplo, no nível i, pode ser refinado em vários *RCs* no nível i +1. A abordagem L'Ecritoire propõe uma estratégia para refinar *RCs*. Essa estratégia consiste em encontrar objetivos no nível i + 1, assumindo que cada interação em um cenário no nível i é considerada como um objetivo a ser obtido no nível i + 1. Por exemplo, um chunk de requisito denominado de RC1, pode ser refinado por vários chunks de requisitos, tais como: RC1.1, RC1.2 e assim por diante, os quais são originados das ações contidas no cenário do RC1 do nível i. Para auxiliar neste processo de refinamento, classificam-se os *RCs* em um dos níveis *Contextual, Funcional e Físico* definidos pela abordagem. Isto facilita o processo de identificação e refinamento e, conseqüentemente, o processo de elicitação de requisitos. Tipicamente, inicia-se pela descoberta de objetivos de projeto (nível contextual), que satisfazem objetivos organizacionais conhecidos, e evolui-se através do refinamento, para a descoberta de objetivos e conseqüentemente *RCs*, nos níveis funcional e físico.

Em Rolland et al (1998), são descritas as diretrizes definidas na abordagem L'Ecritoire a serem aplicadas no processo de elicitação de requisitos. As diretrizes, basicamente, guiam engenheiros de requisitos no desenvolvimento de *RCs,* utilizando essencialmente os relacionamentos do tipo composição, alternativa e refinamento.

Na figura 7, apresentamos um resumo do processo de elicitação de requisitos em L'Ecritoire, tendo como base *RCs*.

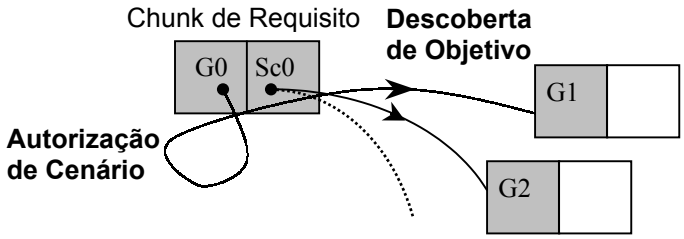

Figura 7. Visão do Processo de Elicitação de L'Ecritoire

Para compreender melhor este processo, observemos a definição do Chunk de requisito (RC1), representando um objetivo e correspondente cenário para um sistema de uma Máquina de Transações Automáticas (*ATM - Automatic Teller Machine*), (figura 8).

Nível 1

RC1

G1:	**Sc1:**
Prover a transação de retirada de dinheiro para nossos clientes do banco, através do uso de uma ATM .	**1. O cliente do banco obtém um cartão do banco;** **2. Então, o cliente do banco retira dinheiro na ATM;** **3. A ATM relata a transação de retirada para o Banco.**

Figura 8. Um exemplo de RC <objetivo, cenário> (Rolland et al. 1998)

Todo processo de elicitação segundo L'Ecritoire, inicia-se a partir da análise de objetivos organizacionais ou de negócio previamente definidos. A abordagem, no entanto, não fornece mecanismos para elicitar estes objetivos e parte do princípio que os mesmos já são conhecidos. No caso do RC1, apresentado na figura 8, o mesmo foi elicitado a partir da análise do objetivo organizacional "*Melhorar os serviços para os nossos clientes do banco"*. Uma das formas possíveis de satisfazer este objetivo é melhorar os serviços, possibilitando que os clientes possam retirar dinheiro de suas

contas usando máquinas de transações automáticas. Define-se o RC1, para representar esta alternativa de projeto que satisfaz o objetivo organizacional em questão. Como o objetivo "*Prover a transação de retirada de dinheiro para nossos clientes, através do uso de uma máquina de transações automáticas (ATM)*", é uma alternativa de projeto, o mesmo é enquadrado no nível contextual de abstração.

O chunk de requisito RC1 é representado contendo o objetivo **G1** e o cenário **Sc1**. Aplicando diretrizes de refinamento ao RC1, podemos descobrir outros objetivos a partir de cada ação do cenário Sc1. Estas ações vão originar objetivos em um nível mais baixo de abstração, tipicamente no nível funcional definido anteriormente. Por exemplo, analisando Sc1, podemos elicitar os seguintes objetivos, os quais deverão ser agrupados em RCs, com seus respectivos cenários associados:

- *G1.1 - Retirar dinheiro de uma máquina ATM em um caminho normal.* O RC1.1, para este objetivo, conteria também o cenário *Sc1.1* associado, descrevendo os passos para a retirada de dinheiro de uma máquina ATM, com sucesso.
- *G1.2 – Obter um cartão do Banco com sucesso.* O RC1.2,para este objetivo, conteria também o cenário Sc1.2, o qual descreveria os passos necessários para obtenção de um cartão pelo cliente do banco, tendo no final sucesso.
- *G1.3 – Relatar transações de retirada para o banco com sucesso.* O RC1.3 , para este objetivo, conteria também o cenário Sc1.3, contendo uma descrição dos passos para relatar as transações de retirada de dinheiros para o banco.

Nos três RCs acima, as exceções ou alternativas para o objetivo relacionado, poderiam ser adequadas em outros RCs. Os chunks de requisito, que possuem exceções ou alternativas, podem ser relacionados através de relacionamentos do tipo OR, indicando que um dos pares relacionados pode ser executado, dependendo de uma

situação específica. No exemplo acima, podemos também encontrar o relacionamento o tipo AND. Os objetivos G1.1 e G1.2, são exemplos de chunks que devem ser relacionados pelo relacionamento AND. Isto ocorre porque, para "Retirar dinnheiro de uma máquina ATM em um cainho normal" (G1.1), é necessário que o cliente possa "Obter um cartão do Banco com sucesso" (G1.2). Como visto anteriormente, o relacionamento de composição (AND) estabelece a ligação entre RCs, que necessitam um dos outros para definir uma funcionalidade completa para o sistema.

Na figura 9, podemos visualizar uma visão geral da abordagem em L'Ecritoire. A figura representa um "mapa" do processo de elicitação, no qual os nós representam as intenções no processo, e as conexões entre os nós são nomeadas com estratégias para obter as intenções.

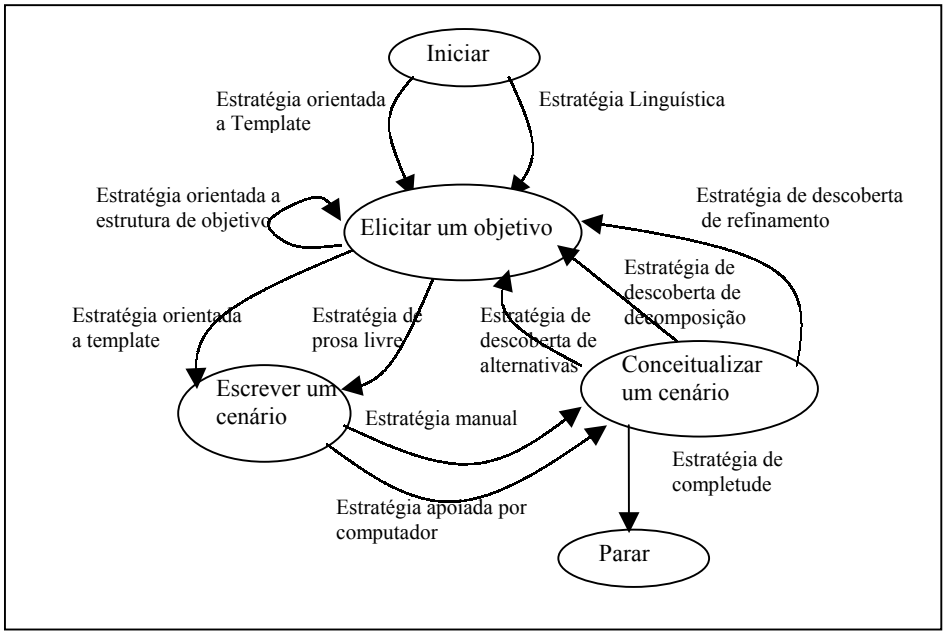

Figura 9. Uma visão geral da Abordagem *Crews- L´Escritoire* (Tawni et al. 1999).

Pode-se entender melhor a figura 9 se a visualizarmos, tendo em vista um mapeamento do tipo <**intenção fonte, intenção alvo, estratégia adotada>.** Neste mapeamento, a estratégia define o caminho que será seguido para alcançar a intenção alvo no processo de elicitação de requisitos. Por exemplo, o mapeamento <Elicitar um

Objetivo, Escrever um Cenário, Estratégia de Prosa Livre > define que para escrever um cenário a partir de um objetivo elicitado, será usado a estratégia de descrição de cenários em prosa livre, sem seguir rigorosos modelos na escrita. No entanto, poderíamos também utilizar a estratégia orientada a *templates* para escrever cenários, que seria representado através do mapeamento <Elicitar um Objetivo, Escrever um Cenário, Estratégia Orientada a Template>. Para ambas as estratégias, a abordagem L'Ecritoire define diretrizes que podem guiar os engenheiros de requisitos na escrita de cenários, utilizando linguagem textual, contendo estruturas lingüísticas previamente definidas. Estas estruturas, bem como uma descrição das estratégias contidas na figura 9, podem ser encontradas em Rolland (1998)

É importante também destacar que, uma ferramenta para a abordagem L'Ecritoire foi desenvolvida (Tawni et al. 1999) e que as pesquisas, visando melhorar a abordagem, bem como a ferramenta de suporte, estão em andamento.

Contudo, a abordagem Crews L'Ecritoire, não trata especificamente e explicitamente da relação de objetivos e requisitos organizacionais com cenários sendo desenvolvidos. Além disso, todo o processo de descoberta de objetivos e cenários ainda depende, em grande parte, da experiência de engenheiros de requisitos em conseguir iniciar o processo de elicitação destes elementos, a partir de interações com usuários. Não são fornecidas heurísticas ou mecanismos que permitam que engenheiros de requisitos estejam familiarizados com o ambiente organizacional, nem de como objetivos de negócio e/ou organizacionais podem originar objetivos funcionais de usuários em relação ao sistema pretendido. Isto ocorre, porque a técnica não define a necessidade de modelagem organizacional, nem aponta diretrizes que possam auxiliar engenheiros de requisitos em como iniciar o trabalho de descoberta de objetivos e cenários.

Assim, consideramos que o processo de descoberta de objetivos e cenários em CREWS pode ser melhorado e complementado, se inicialmente modelarmos requisitos organizacionais através de i*. Uma alternativa seria utilizar as diretrizes que propomos nesta tese (descritas no capítulo 5), juntamente com as já fornecidas pela técnica L'Ecritoire. Poderíamos assim evoluir para a descrição dos cenários de sistemas de software de forma mais sistemática a partir dos modelos organizacionais em i*. Desta forma, engenheiros de requisitos teriam um porto de partida para a descoberta de objetivos e cenários, bem como poderiam relacionar requisitos organizacionais e requisitos funcionais de um sistema pretendido.

3.3 O método GBRAM

O método GBRAM (Goal-Based Requirements Analysis Method) (Anton 1997), lida com os aspectos críticos associados com o processo de descoberta de objetivos de sistemas computacionais. Argumenta-se que a tarefa de identificar objetivos de alto nível é fundamental para o processo de análise de requisitos. GBRAM assume que objetivos de sistemas não têm sido previamente documentados ou explicitamente elicitados a partir dos stakeholders e que analistas devem trabalhar a partir de várias fontes de informação disponíveis, cada uma com seu próprio escopo de conhecimento, visando determinar os objetivos do sistema desejado. O método também suporta a elaboração de objetivos para representar o sistema desejado. Nesta seção, apresentamos uma visão geral do método, diferenciando as atividades de análise e refinamento de objetivos, conforme proposta original. Uma apresentação mais detalhada de como aplicar o método a partir da identificação inicial de objetivos e então traduzir estes objetivos em requisitos operacionais, é apresentada em Anton (1997).

3.3.1 Estratégia de Análise e Refinamento no GBRAM

A figura 10 mostra as atividades que o analista estará envolvido quando aplicar a abordagem GBRAM. As atividades de ***análise*** de objetivos podem ser resumidas, conforme segue:

- atividades envolvidas no passo *Explorar* requerem a investigação da informação disponível;
- atividades envolvidas no passo *Identificar* requerem a elicitação dos objetivos e seus agentes responsáveis, a partir da informação disponibilizada para os analistas a respeito do sistema;
- atividades envolvidas no passo *Organizar* englobam a classificação dos objetivos e a organização destes objetivos de acordo com relações de dependência.

Por sua vez, as atividades de ***refinamento*** de objetivos podem ser resumidas, conforme segue:

- atividades do passo *Refinar* envolvem a eliminação de objetivos redundantes e a junção de objetivos sinônimos;
- *elaborar* refere-se ao processo de análise dos objetivos visando identificar obstáculos de objetivos, bem como construir cenários[4] para descobrir objetivos e requisitos ainda não descobertos;
- *operacionalizar* refere-se ao mapeamento de objetivos para requisitos operacionais visando a especificação final de requisitos.

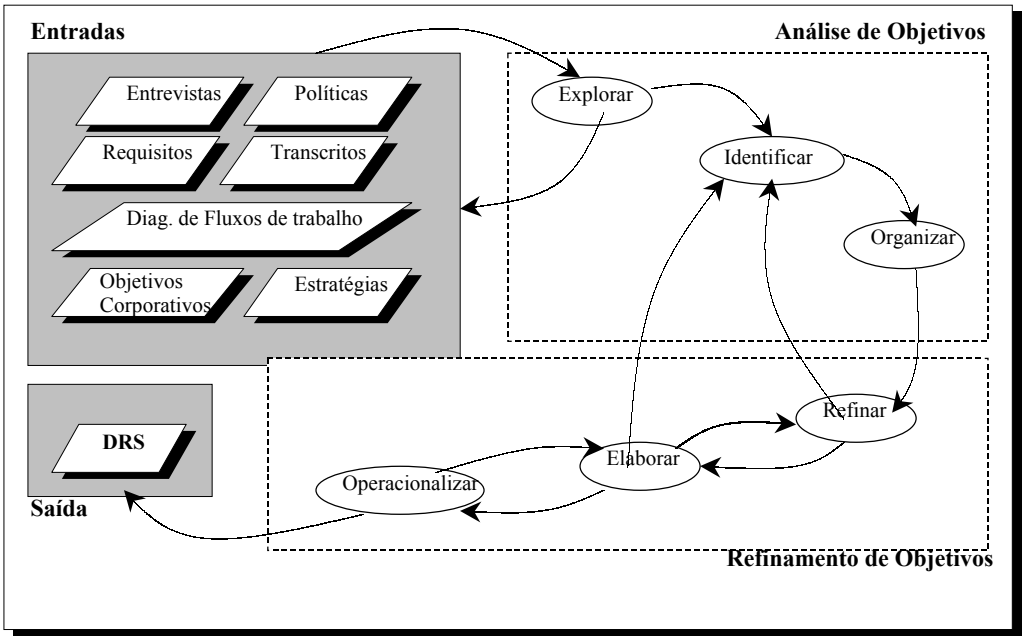

Figura 10. Visão Geral das Atividades no GBRAM

O retângulo no canto superior esquerdo da figura 10 contém as possíveis entradas, as quais podem variar de acordo com a documentação inicialmente disponível para analistas. A saída do GBRAM, (como mostrado na figura 10), é sempre um documento de requisitos de software[5] (DRS). O DRS inclui os requisitos funcionais e não funcionais e deveria ser bastante específico em relação ao comportamento externo do sistema.

O nível de detalhes ou generalidade dos requisitos definidos no DRS é de responsabilidade dos autores do documento. *Stakeholders* ou clientes potenciais tendem a expressar requisitos em termos gerais, enquanto que analistas ou desenvolvedores tendem a expressar requisitos em um nível maior de detalhes, confiando em métodos e notações mais formais e precisas. Através do uso de notações mais formais para reduzir

[4] Cenários para Anton (1997) são descrições comportamentais de um sistema e seu ambiente derivadas de situações específicas.
[5] Um documento de requisitos de software (DRS) é um documento que contem uma descrição completa de "o que" o software fará sem se preocupar "como" isto será feito (Anton 1997)

o nível de ambigüidade, analistas, (os quais podem não estar familiarizados com o domínio de aplicação), produzem documentos que muitas vezes, são incompreensíveis por stakeholders não familiarizados com essas notações. Por outro lado, *stakeholders* são tipicamente especialistas no domínio de aplicação, mas não são treinados em métodos de análise mais precisos. Como já discutido no capítulo 1, requisitos expressos por usuários e clientes tendem a ser ambíguos. É neste aspecto que se ressalta a principal contribuição do método GBRAM (Anton 1997). O DRS produzido, usando GBRAM, ataca o problema comum de ambigüidade de requisitos e por outro lado, também não utiliza notações matemáticas ou formais. GBRAM permite que requisitos sejam expressos em linguagem natural, de forma que estes possam ser facilmente compreendidos por não especialistas em computação, assim encorajando o envolvimento ativo dos stakeholders ao longo do processo.

Para auxiliar no processo de utilização do GBRAM, Anton (1997) tem definido uma série de heurísticas que podem ser usadas por analistas. A seleção de heurísticas a serem utilizadas depende do tipo do sistema desejado e da informação disponível para o analista. São definidos quatro tipos de heurísticas conforme segue:

- heurísticas de identificação;
- heurísticas de classificação;
- heurísticas de refinamento;
- heurísticas de elaboração.

Heurísticas de identificação auxiliam analistas na identificação de objetivos, stakeholders, agentes e restrições a partir de múltiplas fontes. Heurísticas de refinamento empregam uma série de questões e técnicas para reduzir o tamanho do conjunto de objetivos. Heurísticas de elaboração lidam com a necessidade de adquirir

informação mais detalhada, considerando relacionamentos de dependência de objetivos, sugerindo obstáculos de objetivos para os quais cenários deveriam ser construídos, bem como quais cenários deveriam ser elaborados.

É importante ressaltar que entre as várias fontes de origens de objetivos para sistemas computacionais, GBRAM aponta cenários como sendo uma fonte bastante efetiva para a descoberta de novos objetivos, bem como para a construção de novos cenários (Anton 2001). Destacamos a seguir, algumas das heurísticas de identificação e elaboração gerais propostas em GBRAM, mas que podem ser aplicadas na *identificação e elaboração* de cenários e *objetivos*. Uma aplicação destas heurísticas pode ser vista em (Anton 2001).

1. Heurísticas de Identificação de Objetivos– HIG

HIG 1: Mecanismos de abstração podem ser empregados para extrair objetivos da documentação disponível fazendo os seguintes questionamentos:

(a) Qual objetivo(s) esta sentença/frase exemplifica?

(b) Qual objetivo(s) esta sentença/frase bloqueia ou obstrui?

Se a resposta para uma destas questões é positiva, então expresse a sentença/frase como um objetivo que representa o estado que é desejado ou obtido dentro do sistema.

HIG 2: Objetivos podem ser descobertos investigando através dos seguintes questionamentos, as dependências de objetivos para os objetivos previamente especificados:

(a) Quais são as pré-condições deste objetivo?

(b) Quais são as pós-condições deste objetivo?

Como as pré-condições e pós-condições são expressas na forma de objetivos em GBRAM, torna-se possível identificar novos objetivos ainda não descobertos.

HIG 3: Objetivos são também identificados investigando-se os obstáculos[6] para os objetivos já descobertos.

HIG 4: Objetivos podem ser identificados investigando-se possíveis cenários.

HIG 5: Objetivos podem também ser identificados investigando-se restrições[7].

2. Heurísticas de Elaboração de Objetivos via Cenários – HES

HES 1: Um caminho efetivo para identificar cenários candidatos a serem construídos é considerar cada objetivo e investigar: O que acontece se este objetivo não é obtido? Quais são as circunstâncias sob as quais este obstáculo pode ocorrer? Os cenários identificados são elaborados listando as atividades (ações) necessárias para se atingir o objetivo de cada cenário.

HES 2: Outra forma efetiva para identificar cenários candidatos é considerar cada obstáculo e perguntar: Por que este obstáculo ocorreu? Por que este objetivo não foi obtido? Sob quais circunstâncias este obstáculo ocorreria?

HES 3: Os cenários que analistas deveriam tratar com uma atenção especial são aqueles que violam objetivos ou obstáculos.

3. Heurísticas de Elaboração de Objetivos via Obstáculos - HEO

HEO 1: Existe pelo menos um obstáculo de objetivo para todo objetivo. Este é informalmente referenciado como obstáculo trivial e formalmente referenciado como o primeiro caso normal de obstáculo de objetivo. Estes obstáculos são derivados negando-se o verbo no nome do objetivo.

[6] Obstáculos de um objetivo impedem ou bloqueiam a obtenção do objetivo.
[7] Uma restrição de um objetivo estabelece uma condição para a obtenção do objetivo.

HEO 2: Uma sentença que ilustra uma condição a qual impede a obtenção de um objetivo ou ilustra um exemplo de um objetivo sendo bloqueado por outro objetivo, representa um indicativo de um obstáculo e deveria ser expressa como um obstáculo.

HEO 3: Um caminho efetivo para identificar obstáculos de objetivos é considerar cada objetivo e realizar os seguintes questionamentos: De qual outro objetivo(s) ou condição(ções) este objetivo depende? Qual outro objetivo(s) deve ser completado ou obtido para que este objetivo possa ser obtido? Qual objetivo(s) depende deste objetivo? Qual objetivo(s) é uma seqüência ou deve seguir a partir deste objetivo? Pode a falha na obtenção de outro objetivo bloquear a obtenção deste objetivo? Se este objetivo é bloqueado, quais são as conseqüências? A resposta para os questionamentos acima deve ser escrita enfatizando um estado verdadeiro, assim representando adequadamente um obstáculo de objetivo.

HE0 4: Um obstáculo de falha de pré-requisito ocorre quando um objetivo que tem um relacionamento de precedência é obstruído, porque o objetivo de precedência falha. Falhas de pré-requisitos são identificadas considerando cada objetivo e perguntando. De qual outro objetivo(s) este objetivo depende ?

Algumas destas heurísticas serão utilizadas para complementar a nossa proposta de integração de modelos organizacionais e casos de uso. No entanto, como vimos, GBRAM é um método que tem como foco a identificação de objetivos e a organização de requisitos de sistemas ao redor destes objetivos. Como é um método bastante geral, não lida, por exemplo, com as especificidades de uma determinada notação ou diagrama que possa levar a objetivos e cenários. Isto torna bastante difícil o uso do método com técnicas com um rico poder de representação tais como modelos em i*. Outro aspecto a ser considerado é que a GBRAM apenas classifica objetivos em dois tipos ("*Achievement*" e "*Maintenance*"), não lidando com uma questão importante que é a

definição do nível de abstração do objetivo. Objetivos tipicamente podem ser enquadrados em vários níveis de abstração (Cockburn 2000). Isto dificulta, por exemplo, estabelecer um elo de ligação entre objetivos descobertos em GBRAM e objetivos de casos de uso na visão de Cockburn (2000).

3.4 Proposta Enfocando Cenários como meio de melhorar uma Baseline de Requisitos

Segundo Leite et al. (1997), um cenário é uma descrição de situações em um ambiente. Cenários podem ser utilizados como ferramenta para definir "*baselines*" de requisitos. Uma "*baseline*" é uma estrutura que incorpora descrições a respeito do sistema desejado em dado Universo de Discurso[8]. Adota-se que os requisitos de um sistema evoluem tanto ao longo do processo de desenvolvimento quanto na fase de manutenção. Para controlar e conseguir avaliar melhor os impactos decorrentes dessa evolução, é apresentada uma estratégia centrada em cenários. Faz-se uma revisão das várias formas de apresentação de cenários e apresenta-se novos conceitos e elementos que segundo Leite, contribuem no desenvolvimento de cenários mais efetivos e não ambíguos.

Os seguintes conceitos e elementos são apresentados:

- um cenário inicia pela descrição de situações no macrosistema. Considera-se, inicialmente, as interfaces do macrosistema e então descreve-se as interfaces do software com o macrosistema. Neste ponto, a abordagem de Leite et al. (1997), diferencia-se da abordagem de casos de uso, por considerar uma visão mais abrangente, a qual inclui na descrição de cenários, informações sobre o contexto no qual o sistema está inserido;

- um cenário evolui juntamente com o processo de desenvolvimento de software;
- cenários são naturalmente ligados a um LEL (Léxico Estendido da linguagem) e uma VMB (Visão de Modelo Básico) do "*baseline"* de requisitos;
- um cenário descreve situações, com ênfase na descrição comportamental. Cenários usam uma descrição em linguagem natural como representação básica.

Dentre estes conceitos, destaca-se a proposta do desenvolvimento de um léxico estendido da linguagem, o qual é um metamodelo projetado para ajudar na elicitação da linguagem usada no macrosistema. Neste léxico, são descritos todos os termos importantes e que ajudam na compreensão do ambiente do sistema. Fundamenta-se na idéia de que é conveniente compreender a linguagem do problema sem se preocupar em compreender o problema em si.

Este léxico é composto basicamente pelo nome do termo, por uma noção a respeito do termo, bem como de respostas comportamentais relacionadas com o termo, descrevendo informações relacionadas com o contexto no qual o termo está inserido. A figura 11 apresenta um exemplo de entrada no Léxico Estendido de Linguagem para um domínio de emissão de passaportes na Argentina (Leite et al. 1997).

O nome do termo é **Cabine de Fotos** e a noção do mesmo define o que ele representa no contexto do domínio de emissão de passaportes. As respostas comportamentais expressam as ações que podem ocorrer na Cabine de Fotos no

[8] Segundo Leite (Leite et al. 1997), Universo de Discurso é todo o contexto no qual o software será desenvolvido e operado.

processo de emissão de passaportes. As palavras em negrito, na figura 11, representam termos definidos em outras entradas do LEL para o mesmo domínio.

Tendo como auxílio o LEL, são desenvolvidos os cenários para o sistema. Cada cenário consiste basicamente dos seguintes elementos:

- **título:** um título para o cenário;
- **objetivo:** um objetivo a ser obtido pelo cenário;
- **contexto:** descreve o contexto do cenário e um estado inicial importante;
- **recurso:** suporte ou dispositivos que devem estar disponíveis no cenário;
- **ator:** uma pessoa ou uma estrutura organizacional que possui uma regra no cenário;
- **episódios:** uma série de sentenças que detalham e descrevem o comportamento do cenário.

Cabine de Fotos

- Noção:
  - Um setor de **Documento e Divisão de Certificados**.
  - Lugar onde a foto do **solicitante do passaporte** é tirada e os encargos cobrados.
- Resposta Comportamental:
  - O **formulário** é impresso com o mesmo número existente na foto.
  - O **solicitante do passaporte** recebe duas cópias da foto.
  - O encarregado da **Cabine de Fotos** arquiva a terceira cópia.

Figura 11. Exemplo de entrada no LEL: Cabine de Fotos.

Alguns termos, tais como: ator, recursos, nomes, e outros, podem ser obtidos diretamente do LEL. No termo episódios, define-se todos os episódios envolvidos no cenário. Se for necessário refinar e descrever melhor um destes episódios, o mesmo pode ser encarado como subcenário e descrito novamente como se fosse um novo cenário. Os termos ***Restrição*** e ***Exceção*** são usados para respectivamente, **restringir** a

execução de um episódio em algum aspecto e definir uma **exceção** que pode vir a interromper o episódio.

A figura 12 apresenta um exemplo de descrição de cenário cujo título é organizar uma reunião (Leite et al. 1997).

TÍTULO:
Organizar uma reunião.
OBJETIVO:
Garantir que a reunião será realizada eficientemente.
CONTEXTO:
A reunião deve acontecer após o projeto de agendamento da reunião.
ATORES:
Agendador;
Secretária;
Participantes;
RECURSOS:
Equipamentos e materiais de escritório
Temas da reunião
Lista de Participantes
Agenda
Mídias de Comunicação(telefone,fax, computador)
EPISÓDIOS:
O agendador fornece instruções para a secretária a respeito do convite para a reunião.
Chamada para a reunião. *Exception :* Cancela uma reunião. Muda uma data da reunião, Muda os requisitos da reunião.
Informa presença.
Informa não presença.
Requer Equipamentos e Materiais de escritório.
Confirma a reunião com participantes.
A Secretária confirma que o equipamento e materiais de escritório estão disponíveis para a data da reunião.
A Secretária confirma que o espaço físico está disponível para a reunião. #

Figura 12. Um exemplo de Descrição de Cenário em Leite et al. (1997).

Este exemplo mostra os elementos necessários que devem ser definidos em um sistema, que tem como um dos objetivos, realizar e programar reuniões em uma empresa. Os termos sublinhados indicam que esses termos estão definidos no LEL (Léxico Estendido da Linguagem). O cenário é organizado com os elementos apresentados anteriormente. O elemento **Episódios** é o corpo do cenário e descreve-se

no mesmo, os episódios possíveis para este cenário. Estes episódios formam a seqüência de passos necessária à obtenção do objetivo do cenário.

Nesta abordagem, verifica-se que um cenário pode ter mais do que um episódio, o qual por sua vez, pode ser considerado como um caso específico de cenário e definido estruturalmente como tal. O principal enfoque adotado nesta abordagem reside na utilização do LEL para auxiliar no desenvolvimento de cenários. Argumenta-se que, desta forma, o trabalho de construção de cenários é facilitado, pois os principais termos do domínio de aplicação são claramente compreendidos e formalmente definidos. Assim, cenários podem ser mais facilmente integrados em uma "*baseline*" de requisitos que possibilita a evolução de requisitos com um controle e rastreamento adequados.

Esta proposta possui algumas características particulares, tais como: associar objetivos a cenários e não a atores, permitir definir objetivos de cenários como objetivos de negócio, objetivos pessoais ou de usuário, etc, bem como descrever o contexto de um cenário para representar e delimitar a atuação geográfica do mesmo. No entanto, acreditamos que é importante que engenheiros de requisitos possam também visualizar em um nível macro e em uma representação específica, todas as informações importantes sobre ambiente organizacional e principalmente como os atores do ambiente organizacional estão relacionados diretamente a objetivos de cenários. Uma representação gráfica de modelos organizacionais tal como na técnica i*, poderia auxiliar nesta tarefa, mostrando os relacionamentos entre os atores do sistema, com suas motivações, objetivos e intenções no ambiente organizacional, bem como suas expectativas sobre sistemas computacionais pretendidos. A integração de i* com esta abordagem de cenários poderia facilitar o trabalho de identificação de objetivos, bem como a descrição dos cenários.

3.5 Considerações Finais

Investigamos neste capítulo, algumas técnicas baseadas em cenários integradas com abordagens orientadas a objetivos. A principal vantagem advinda com uso destas técnicas é facilitar a comunicação entre os diversos *stakeholders* visto que a parceria de objetivos e cenários torna o processo de elicitação e documentação de requisitos mais consistente, completo e menos propenso a ambigüidades. Outro aspecto positivo é que a descrição de cenários e objetivos é algo bastante natural que realizamos ou observamos no dia a dia. Esta visão, adaptada à descrição de interações entre usuários e sistemas de software, torna menos árduo o trabalho de engenheiros de requisitos. Entre as técnicas apresentadas neste capítulo, destaca-se a técnica de Caso de Uso observada sob a ótica de Cockburn (2000), sendo a mesma também integrante e considerada um elemento chave na linguagem de modelagem visual UML (Booch et al. 1999)

Contudo, é necessário salientar que um dos maiores problemas com as técnicas descritas neste capítulo, é que em geral, as mesmas não tratam e nem consideram adequadamente os requisitos organizacionais previamente definidos pela organização. Além disso, definir inicialmente os objetivos/cenários para sistemas computacionais é um trabalho árduo que exige um alto grau de experiência de desenvolvedores. Utilizar e investigar uma fonte de requisitos na busca de objetivos/cenários que expressem a utilização de sistemas ainda é uma tarefa que, na maioria das vezes, é desprovida de heurísticas e diretrizes que possam auxiliá-la.

Por outro lado, vemos também que muitas informações presentes em modelos organizacionais que poderiam ser utilizadas para auxiliar e guiar as descrições de requisitos funcionais em cenários, não são consideradas de forma efetiva. Engenheiros de requisitos que conhecem, compreendem e utilizam as informações sobre o ambiente organizacional, têm uma probabilidade muito menor de gerar documentos de requisitos

inconsistentes, bem como podem descrever documentos de requisitos mais completos, por saberem como estes requisitos estão realmente associados aos objetivos da organização e/ou de usuários.

No próximo capítulo, descreveremos algumas técnicas de modelagem organizacional, com ênfase no *framework* i*, concluindo assim a descrição dos elementos necessários à definição da nossa proposta de integração de i* com Casos de Uso.

Capítulo 4

Modelagem Organizacional

Há um consenso que para desenvolver sistemas de software de qualidade torna-se imprescindível conhecer e compreender o ambiente organizacional, no qual o software estará inserido e será executado. Uma das metas da Engenharia de Requisitos consiste em elicitar e especificar da melhor forma possível os requisitos para um novo sistema de software. No entanto, softwares visam fundamentalmente satisfazer ou colaborar na obtenção de metas e objetivos previamente definidos pela organização. Isto implica em afirmar que o sucesso de sistemas de software depende, entre outros aspectos, do grau de satisfação dos objetivos e estratégias organizacionais. Além disso, o trabalho de engenheiros de requisitos e demais desenvolvedores é facilitado a partir do momento em que há um pleno conhecimento do ambiente organizacional. Este conhecimento inclui: comportamento, motivações, intenções e objetivos dos atores da organização, bem como descrições técnicas dos processos organizacionais e de que forma atores participam, integram e conduzem estes processos.

Várias técnicas têm sido propostas com o intuito de modelar ambientes organizacionais. Algumas destas técnicas estão relacionadas com o entendimento, elicitação e documentação de regras de negócio (Leite et al. 1998) (Leonardi et al. 1998) (Rosca et al. 1997), processos de negócio (Fiorini et al. 1995) (Fiorini et al. 1996) (Fiorini 1999) (Estrada et al. 2002) bem como de fluxos de trabalho (*workflow)* (Bortoli et al. 2000) (Estrada et al. 2001) (Penadés et al. 2001). Outras técnicas tais como i* (Yu 1995) e Bubenko (1993), além de permitir expressar processos de negócio e suas atividades associadas, permitem também modelar aspectos intencionais e motivações associadas com atores estratégicos da organização, inclusive a respeito de possíveis

sistemas computacionais pretendidos pela organização. Técnicas tais como i* possibilitam-nos expressar melhor os "porquês" que estão associados a práticas e estruturas organizacionais existentes. A captura deste tipo de conhecimento facilita, principalmente, a tomada de decisões por parte de atores da organização, sobre possíveis mudanças nos processos de negócio, que permitam obter as metas estratégicas da organização da melhor forma possível. Estas mudanças em processos de negócio normalmente incluem o desenvolvimento de sistemas computacionais que apóiam a execução de atividades da organização.

Em geral, pretende-se com o uso de técnicas de modelagem organizacional, representar o conhecimento a respeito da organização e propiciar que este conhecimento seja utilizado e integrado efetivamente no processo de Engenharia de Requisitos. No entanto, alguns problemas ainda persistem nessa integração. Não existem caminhos sistemáticos que permitam que as informações capturadas sobre o ambiente organizacional, tais como: atores e objetivos dos mesmos em relação a processos de negócio e sistemas de software pretendidos pela organização, possam ser mapeadas de alguma forma em descrições funcionais e não funcionais de sistemas computacionais. Entendemos que integrar modelos organizacionais com técnicas que descrevem requisitos de software, tais como casos de uso, é um aspecto importante que pode efetivamente resultar em softwares de mais qualidade. Como já foi dito anteriormente (veja capítulo 3), cenários em geral, não consideram satisfatoriamente requisitos organizacionais. Consideramos, que a partir da integração de modelos organizacionais e cenários, engenheiros de requisitos podem elicitar e documentar requisitos de sistemas, estabelecendo uma relação direta entre estes requisitos e os objetivos previamente definidos pela organização. Entre outras vantagens advindas desta integração, engenheiros de requisitos poderiam também facilmente identificar e evitar a descrição

de requisitos não necessários, bem como utilizar e mapear para requisitos funcionais, as informações, em modelos organizacionais, sobre intenções da organização em relação a sistemas computacionais pretendidos. Estes aspectos constituem as principais metas da nossa tese.

Neste capítulo, apresentamos uma visão geral das principais técnicas de modelagem organizacional, e em seguida, no capítulo 5, descrevermos a nossa proposta de integração deste tipo de modelagem com a técnica baseada em cenários já bastante conhecida e denominada de Caso de Uso. Nas seções 4.1, apresentamos uma descrição das principais características da técnica de modelagem organizacional i* (Yu 1995), enquanto na seção 4.2, descrevemos as principais características da técnica de Bubenko (1993). Já na seção 4.3, apresentamos os principais aspectos de modelagem organizacional envolvidos no processo de desenvolvimento de software RUP (Rational Unified Process), bastante utilizado no meio industrial (Kruchten 2000), (www.rational.com). Na seção 4.4, apresentamos as considerações finais do capítulo.

4.1 A técnica i*

A técnica i* objetiva possibilitar a representação/modelagem de aspectos organizacionais envolvidos com processos (Yu 1995). Basicamente, permite descrever aspectos de intencionalidade e motivações envolvendo atores em um ambiente organizacional. Do ponto de vista da Engenharia de Requisitos, i* auxilia na compreensão do ambiente organizacional (incluindo potenciais sistemas de software), bem como permite explorar propostas alternativas de sistemas mostrando como o ambiente de trabalho, envolvendo atores da organização (processos de negócio), seria afetado e modificado para atender e suportar a implantação de alguma proposta alternativa. Assim, i* pode auxiliar na importante tarefa atribuída a engenheiros de

requisitos, que é a de avaliar os impactos que a introdução de novos sistemas podem causar nos participantes (atores) da organização, como também nos clientes destas organizações. Basicamente, esta técnica provê uma compreensão das razões ("Porquês") que estão associadas aos requisitos de software. Para descrever o ambiente organizacional, i* propõe dois modelos: O Modelo de Dependências Estratégicas (**SD**) e o Modelo de Razões Estratégicas (**SR**), os quais são descritos a seguir.

4.1.1 Modelo de Dependências Estratégicas.

O Modelo de Dependências Estratégicas é composto por **nós** e **ligações**. Os nós representam os **atores** no ambiente e as ligações são as **dependências** entre os atores. Por **ator** entende-se uma entidade que realiza ações para obter objetivos no contexto do ambiente organizacional. Atores dependem uns dos outros para obter objetivos. O ator que depende de alguma forma de outro ator é chamado de ***Depender*** e o ator que atende e satisfaz o *Depender* é denominado de ***Dependee.*** O objeto ou elemento de dependência entre *Depender* e *Dependee* é denominado de ***Dependum***. Portanto, haverá relacionamentos do tipo *Depender* $\rightarrow$ *Dependum* $\rightarrow$ *Dependee*.

As dependências apresentadas neste modelo podem ser de diferentes tipos, tendo como base o tipo do *Dependum*. A seguir descrevemos os quatro tipos de dependências propostos em i*:

- dependência de **objetivo**: o *Depender* depende do *Dependee* para modificar um estado do mundo. O *Dependee* tem como papel satisfazer/alcançar um objetivo para o *Depender*. O *Dependee* pode obter este objetivo e modificar o estado atual escolhendo a forma como isto será feito, tendo liberdade para tomar as decisões necessárias para a obtenção do objetivo. No entanto, o *Depender* torna-se vulnerável, pois o *Dependee* pode falhar em atingir este objetivo.

- dependência de **tarefa**: o *Depender* depende do *Dependee* para executar uma atividade, sendo de responsabilidade do *Depender* mostrar o caminho como isto será feito. No entanto, não é fornecido para o *Dependee* "o porquê" da realização da tarefa. O *Depender* torna-se vulnerável, pois o *Dependee* pode falhar na realização da tarefa não seguindo os passos indicados para realizar a tarefa ou pode estabelecer outras prioridades antes de realizá-la. Neste tipo de dependência, o *Depender* toma as decisões e os objetivos do *Depender* não são de conhecimento do *Dependee*.
- dependência de **recurso**: o *Depender* depende do *Dependee* em relação à disponibilidade de um recurso, seja ele físico ou de informação. Isto significa que o *Dependee* deve fornecer um recurso que o *Depender* necessita para realizar outras atividades no ambiente organizacional. O *Depender* torna-se vulnerável, pois o recurso pode não ser disponibilizado pelo *Dependee*. Neste tipo de dependência os aspectos relacionados com decisões não são considerados, pois o que está em questão é a disponibilidade ou não de um recurso.
- dependência de **objetivo-soft**: define-se objetivo-soft como um objetivo cuja avaliação de realização é bastante subjetiva e o seu significado não é claramente conhecido. Assim não se pode afirmar objetivamente se o mesmo foi satisfeito. Geralmente, o entendimento e avaliação do objetivo-soft acontecem ao longo do processo da realização de tarefas associadas com o objetivo-soft. Estes objetivos-soft são também referenciados como requisitos não funcionais no contexto da Engenharia de Requisitos. O *Depender* depende do *Dependee* esperando que o mesmo realize alguma tarefa que satisfaça o objetivo-soft. A decisão se o objetivo é ou não satisfeito, é tomada pelo *Depender* com base nas

atividades realizadas pelo *Dependee*. O *Depender* torna-se vulnerável, pois o *Dependee* pode falhar na obtenção das condições que satisfaçam ao objetivo-soft.

Para entender melhor os conceitos associados com o modelo de Dependências Estratégicas (SD), vejamos o exemplo apresentado na figura 13, retirado de (Yu 1995). Este exemplo modela as dependências estratégicas envolvidas no processo de agendamento de reuniões em uma organização. Este processo é apoiado por um sistema computacional denominado Agendador de Reuniões. O iniciador de reunião depende do participante em relação ao seu comparecimento à reunião. O iniciador de reunião delega boa parte do trabalho de agendamento de reuniões ao sistema agendador de reuniões. O agendador de reuniões determina quais são as datas possíveis para o agendamento de uma reunião com base na informação de disponibilidade (dependência do tipo tarefa Entra Datas Disponíveis (m)), fornecida por cada participante. O iniciador de reunião não interfere na forma em que o agendador de reuniões determina as datas de agendamento aceitáveis, contanto que as mesmas sejam encontradas e definidas. Isto reflete-se na dependência do tipo objetivo Reunião Agendada do iniciador para o agendador. Por outro lado, para se chegar a uma data de consenso para o agendamento de uma reunião, participantes dependem que o agendador de reuniões forneça a proposta de data (dependência do tipo recurso Data Proposta(m)). Uma vez proposta uma data de agendamento, o sistema agendador de reuniões depende que os participantes concordem com a mesma (dependência do tipo recurso Concordância (m,p)). Para participantes importantes, o iniciador de reunião depende criticamente do comparecimento destes à reunião (dependência do tipo objetivo Comparecer Reunião(ip,m)), bem como também tem o desejo de garantir que estes participantes

importantes compareçam à reunião (dependência do tipo objetivo-soft Assegurado(ComparecerReunião (ip.m))). Finalmente, o agendador de reuniões depende que o iniciador de reunião forneça um intervalo de datas, dentro do qual uma reunião deve ser agendada (dependência do tipo tarefa EntrarIntervaloDatas(m)).

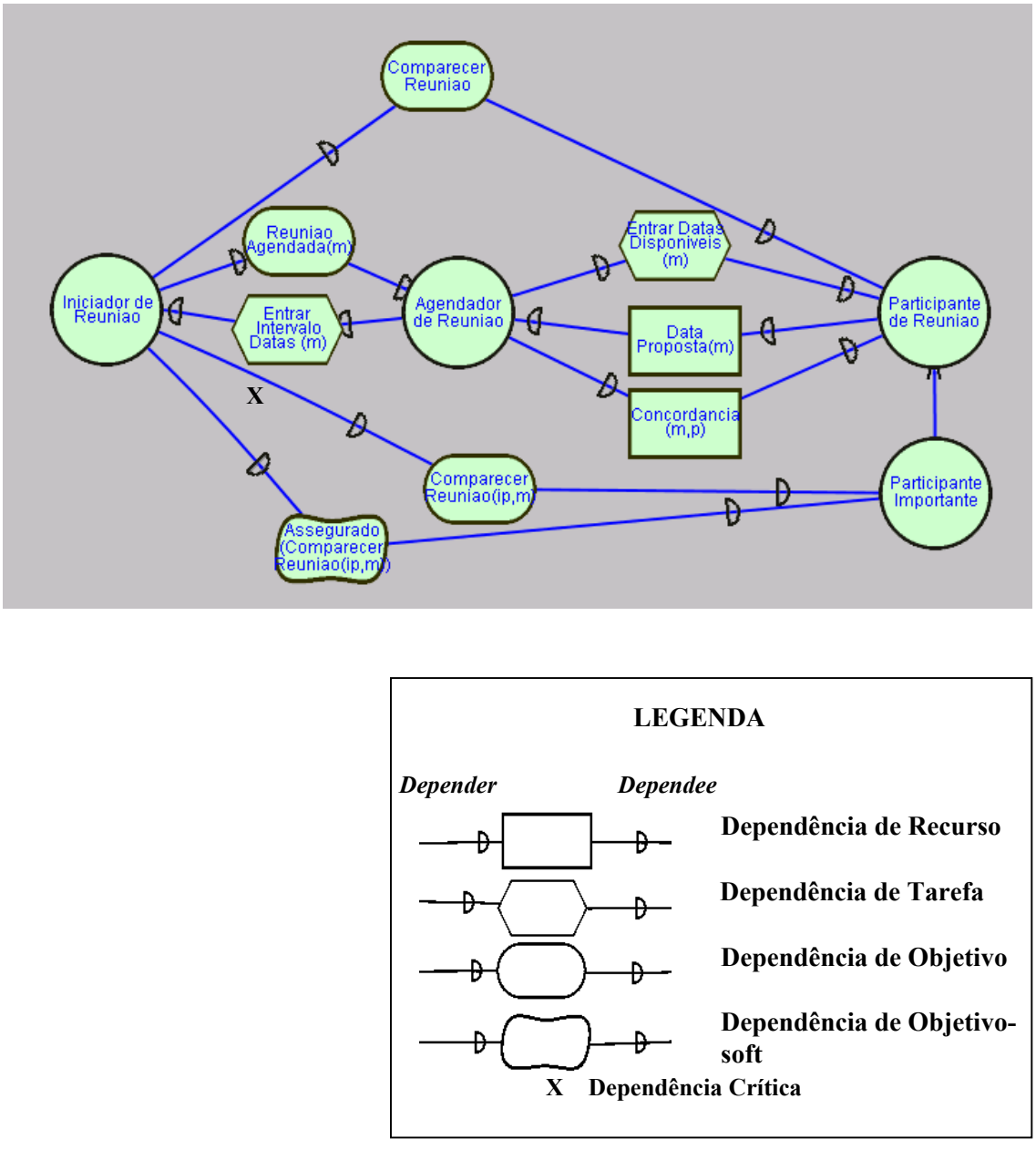

Figura 13. Modelo de Dependências Estratégicas para agendamento de reuniões.

Tendo como base estes elementos definidos para o modelo de dependências estratégicas, tanto as intenções quanto motivações e objetivos organizacionais podem ser modelados e várias alternativas para o desenvolvimento de sistemas computacionais podem ser avaliadas, optando-se por aquela que melhor satisfaça os objetivos de todos os stakeholders.

A seguir, descrevemos resumidamente o outro modelo que compõe a técnica i*, o chamado modelo de Razões Estratégicas (SR).

4.1.2 Modelo de Razões Estratégicas.

O modelo de Razões Estratégicas é um modelo complementar ao modelo de dependências estratégicas apresentado na seção anterior. Este modelo permite compreender e modelar de forma mais detalhada as razões associadas com cada ator e suas dependências. Enquanto o modelo de Dependências Estratégicas prove um nível de abstração, no qual modela-se somente os relacionamentos externos entre atores, o modelo de Razões Estratégicas permite uma maior compreensão a respeito das razões estratégicas de atores em relação a processos da organização e como os mesmos são expressos. O modelo de Razões Estratégicas auxilia no processo de Engenharia de Requisitos, permitindo que elementos de processos e as razões por detrás dos mesmos sejam expressas. Na Engenharia de Requisitos, o modelo de Razões Estratégicas pode ser utilizado para compreender como sistemas estão relacionados/envolvidos em rotinas de atores da organização para gerar alternativas, bem como para modelar e suportar o raciocínio de atores organizacionais a respeito destas alternativas.

O princípio básico para desenvolver este modelo é observar as razões que estão associadas com cada ator em relação aos relacionamentos de dependência com outros atores. Um bom caminho para iniciar a decomposição é observar como os *Dependee*

podem satisfazer os *dependums* associados com os mesmos e a partir desse ponto, observar e decompor as intenções e razões organizacionais estratégicas como um todo.

Basicamente, este modelo é composto de nós e ligações. Os nós no modelo de Razões Estratégicas têm como base os tipos de *Dependum* definidos no modelo de dependências estratégicas**: objetivo, tarefa, recurso e objetivo-soft**. Tem-se, basicamente, dois tipos de classes de ligações: **ligação meio-fim** e **ligação de decomposição de tarefa**. Os mesmos são descritos a seguir:

- **ligação meio-fim (*means-end*):** este tipo de ligação está associada à obtenção de um determinado fim, o qual pode ser um objetivo, recurso, objetivo-soft ou uma tarefa através de um caminho. Este caminho ou meio para obter o fim é geralmente definido em termos de tarefas que são necessárias para atingir/obter o fim desejado.
- **ligação de decomposição de tarefa (*task decomposition*):** uma **tarefa** é modelada em termos de sua decomposição em **sub-componentes** através de uma ligação de decomposição. Permite-se com este tipo de ligação a decomposição em unidades menores de um nó tarefa para representar de forma mais detalhada as razões associadas com a realização da tarefa. Estas decomposições por sua vez podem estar ligadas através de ligações de **dependência** a outros nós pertencentes à decomposição de outros atores. Isto é normalmente necessário, sempre que o relacionamento for relevante para a compreensão das razões e intenções organizacionais.

Neste modelo, uma rotina representa um subgrafo incluindo todas as razões, bem como os meios para se atingir um fim do ponto de vista de um ator. Através deste modelo, pode-se modelar várias alternativas (meios) para se obter fins estratégicos para um ator.

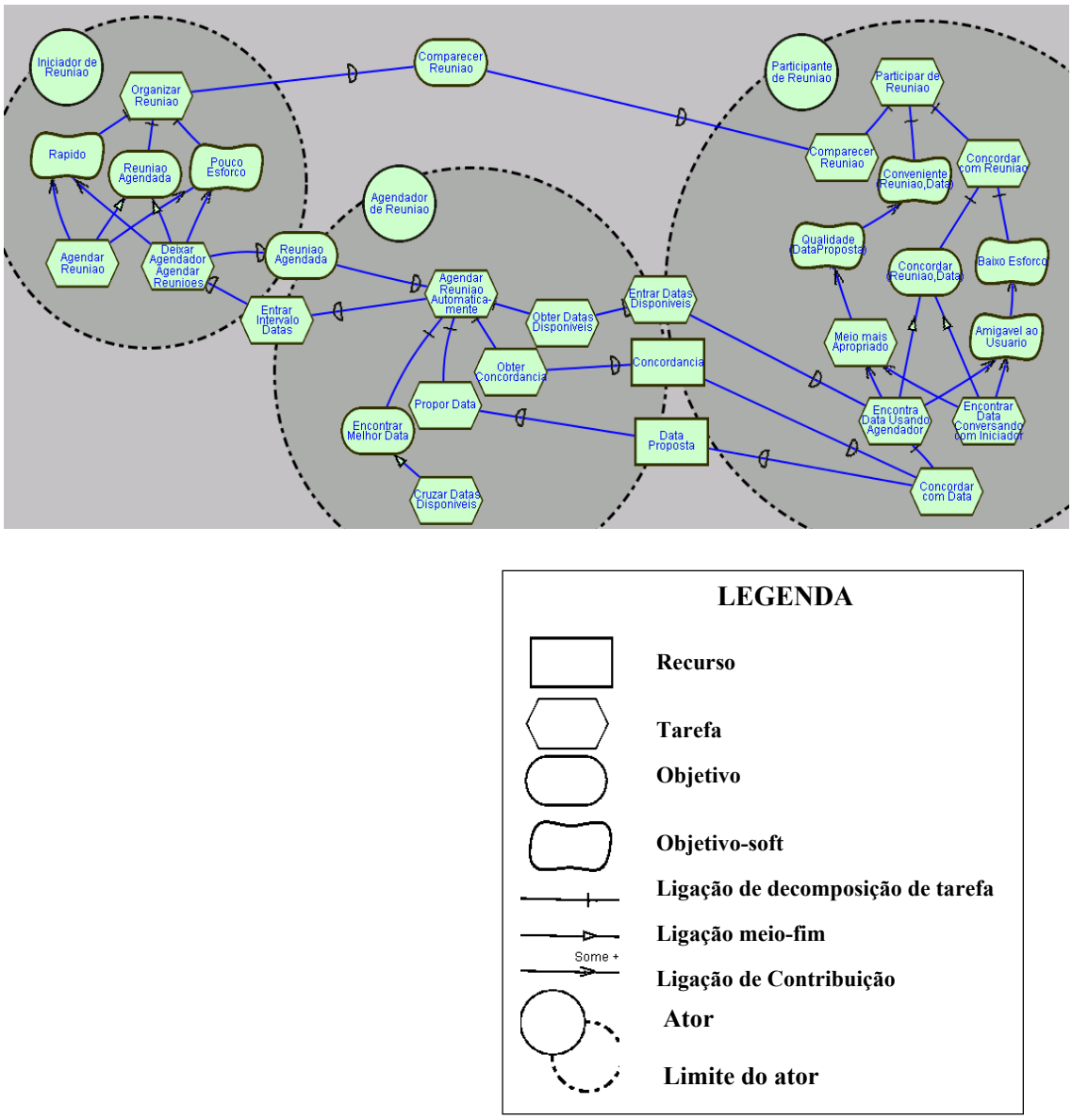

Figura 14. Modelo de Razões de Estratégicas (SR) para o Agendamento de Reuniões, considerando a existência de um sistema computacional para realizar o agendamento.

Apresentamos na figura 14, um exemplo de modelo de razões estratégicas para o problema de agendamento de reuniões. O modelo de Razões Estratégicas para este problema consiste em descrever os relacionamentos intencionais "internos" de atores da organização, tais como ligações meio-fim que relacionam elementos de processos, provendo assim representações explícitas dos "porquês", "comos" e alternativas. Desta forma, para cada ator apresentado no modelo de Dependências Estratégicas da figura

13, descrevemos no modelo de Razões Estratégicas as razões estratégicas sobre processos associados com cada ator.

Observando a figura 14, verificamos que o ator Agendador de Reuniões deve lidar com um pedido para organizar uma reunião. Desta análise, advém a tarefa **Organizar Reunião**. Esta característica é importante antes de decidir, por exemplo, quais aspectos do agendamento de reuniões poderiam ser cobertos por um possível sistema baseado em computador, quais os tipos de reuniões a serem agendadas, etc. Assim, permite-se explorar as alternativas do processo de agendamento.

Podemos também verificar nesta figura, que para realizar uma reunião, o ator Iniciador de Reunião depende do ator Participante de Reunião para que o objetivo comparecer reunião seja satisfeito. O fato da reunião ser agendada é indicada pelo subobjetivo *Reunião Agendada* associado com a tarefa *Organizar Reunião*. Neste ponto, outros subobjetivos poderiam ser incluídos, tais como: equipamentos a serem reservados para a reunião e memorandos a serem enviados, visando relembrar a reunião. Observamos também que a organização da reunião deve ser feita de forma rápida e com menor esforço possível. Estes desejos são representados pelos objetivos-soft *Rápida* e *Pouco Esforço.* O iniciador de reunião pode agendar uma reunião via processo manual (*tarefa Agendar Reunião*) ou utilizando o sistema agendador de reuniões (*tarefa Deixar Agendador Agendar Reunião*). Desta forma, grande parte do trabalho de agendar reunião pode ser atribuído ao ator agendador de reuniões. O ator agendador de reuniões possui a tarefa A*gendar Reunião Automaticamente,* a qual é decomposta em quatro sub-componentes através da ligação decomposição de tarefa: *ObterDatasDisponíveis, EncontrarMelhorData, ProporData e ObterConcordância*. Assim, para agendar uma reunião, o sistema agendador de reuniões obtém datas disponíveis de agendamento dos participantes, encontra melhores datas possíveis de agendamento a partir do cruzamento

das datas fornecidas pelos participantes, propõe uma data específica para o agendamento e finalmente solicita concordância dos participantes em relação a data de agendamento proposta. Por sua vez, para iniciar o processo de agendamento, o iniciador de reuniões precisa fornecer para o sistema agendador de reuniões, um intervalo de datas para agendamento de reunião (dependência do tipo tarefa EntrarIntervaloDatas).

Do ponto de vista do ator Participante de Reunião, pode-se se observar que espera-se dos participantes que os mesmos participem ativamente no processo de agendamento (*tarefa Concordar com Reunião*) e após isto, compareçam (*tarefa Comparecer Reunião*) à reunião. Para um participante, concordar com uma reunião, consiste inicialmente em chegar a uma data conveniente (*Concordar (Reunião, Data*)). Isto requer que o mesmo forneça informações de disponibilidade para o sistema agendador de reuniões e então tente concordar com datas propostas. Participantes também querem selecionar datas convenientes às suas atividades, bem como desejam que o processo de agendamento seja o mais breve e com menor esforço possível. Estes desejos são representados pelos objetivos-soft *Conveniente(Reunião,Data)* e *Baixo Esforço*, respectivamente.

De uma forma geral, a técnica i* tem sido considerada de fácil entendimento pelos diversos *stakeholders*. Além disso, os recursos gráficos da técnica são apontados como suficientes para representar os aspectos mais relevantes do ambiente organizacional (Alencar et al. 1999). Verificamos, no entanto, observando a técnica mais detalhadamente, que a mesma poderia contribuir e ser utilizada de forma mais efetiva no processo de Engenharia de Requisitos, se os requisitos organizacionais representados em i*, fossem relacionados aos requisitos funcionais dos sistemas pretendidos. Neste aspecto, podemos observar nas dependências do tipo objetivo, recurso, tarefa e objetivo-soft, estabelecidas entre os atores em i*, uma rica fonte de

informações para descrever requisitos funcionais e detectar alguns requisitos não-funcionais.

O que propomos é integrar este tipo de técnica com casos de uso, podendo desta forma descrever requisitos funcionais, através do mapeamento das informações representadas em i* para casos de uso. Este tipo de integração, basicamente, permite que engenheiros de requisitos possuam uma fonte de informação para a descrição de requisitos funcionais, bem como possam raciocinar a respeito de como o sistema pretendido irá satisfazer os objetivos da organização, por que ele é necessário, quais as alternativas existentes, quais as implicações das alternativas para as várias partes interessadas, entre outros aspectos.

4.2 A técnica de Bubenko

O desenvolvimento de modelos organizacionais, através da técnica de Bubenko, tem como base a representação do conhecimento da organização em cinco submodelos (Bubenko et al. 1994). Estes submodelos são criados para representar diferentes aspectos do ambiente organizacional, os quais incluem informações relacionadas com atores, atividades, objetivos, conceitos e requisitos organizacionais. A modelagem destas informações permite uma melhor compreensão do ambiente organizacional, bem como pode servir de fonte adicional de conhecimentos a ser utilizada no processo de elicitação de requisitos de software.

Propõe-se na abordagem de Bubenko (1993) (Bubenko et al. 1994), integrar engenheiros de requisitos e usuários no desenvolvimento de modelos organizacionais, estabelecendo dessa forma um elo que pode facilitar a definição de requisitos para novos sistemas de software. Modelos organizacionais também são úteis aos demais desenvolvedores engajados em outras etapas do processo de Engenharia de software.

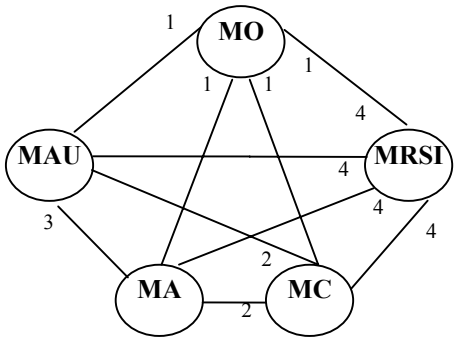

Figura 15. Relacionamento entre os Submodelos na Técnica de Bubenko.

Os cinco submodelos propostos por Bubenko (1993) para modelagem organizacional são: Modelo de Objetivos [MO], Modelo de Atividades e Uso [MAU], Modelo de Atores [MA], Modelo de Conceitos [MC] e Modelo de Requisitos do Sistema [MRSI]. Na figura 15, são apresentados estes submodelos e os relacionamentos entre os mesmos.

Cada um destes submodelos possui um enfoque que trata diferentes aspectos da organização, como segue:

1. **modelo de objetivos (MO):** este modelo descreve de uma forma estruturada, o componente "porquê" de um documento de requisitos. São representadas e analisadas *Objetivos* e *Regras de Negócio*, para uma atividade organizacional específica (ou conjunto de atividades), existentes a serem modificadas ou projetadas. Outros tipos de componentes deste modelo são problemas, causas (de problemas), oportunidades e ações de desenvolvimento. Objetivos e regras são considerados os elementos mais "importantes" neste modelo, sendo que outros tipos de componentes são criados para auxiliar e suportar a descoberta, bem como a formulação dos objetivos e regras. O modelo de objetivos é um grafo contendo os tipos de componentes mencionados representados como nós, conectados por relacionamentos do tipo "motiva" ou "influencia". O relacionamento "motiva" é visto como um mecanismo de

refinamento, no qual, por exemplo, um objetivo é refinado em sub-objetivos. O relacionamento "influencia" é um relacionamento indicando influências positivas e negativas entre os componentes do modelo de objetivos.

2. **modelo de conceitos (MC):** O modelo de conceitos é utilizado na modelagem organizacional para definir conceitos sobre objetivos, relacionamentos, propriedades e outros aspectos importantes do universo da organização. Estas definições são utilizadas para estabelecer um entendimento comum entre usuários, clientes e engenheiros de requisitos e outros stakeholders em relação a conceitos importantes da organização. Um conceito é representado por um "Nome do Conceito" o qual é um elemento do domínio de interesse, sobre o qual deseja-se raciocinar a respeito. Caracteriza-se melhor um conceito através do estabelecimento de relacionamentos com outros conceitos. Um atributo neste modelo é um conceito que é utilizado somente para caracterizar outro conceito. Um Grupo Componente de Modelo de Conceito é um agrupamento de conceitos, relacionamentos e atributos de conceitos. Um relacionamento do tipo Conceito Binário é um relacionamento semântico entre dois conceitos ou entre um e o mesmo conceito. O relacionamento "É-UM" entre conceitos expressa generalização. O relacionamento Parte-de permite relacionar conceitos a partir da definição de um conceito de mais alto nível e outro(s) conceito(s) fazendo parte deste conceito, semelhante à agregação.
3. **modelo de atores (MA):** este modelo é utilizado para analisar e definir o conjunto de atores das atividades da organização, bem como os relacionamentos entre os mesmos. Atores podem ser pessoas, grupos, posições desempenhadas na organização, unidades organizacionais,

máquinas, etc. O modelo de Atores permite uma melhor compreensão dos relacionamentos organizacionais e das possibilidades de comunicação entre os atores da organização. Este modelo reflete a estrutura de papéis e de responsabilidades na realização de tarefas atribuídas na organização e também pode suportar o controle do uso de recursos da organização.

4. **modelo de atividades e uso (MAU):** este modelo descreve as atividades existentes, a serem modificadas ou projetadas em uma organização. Estas atividades são definidas do ponto de vista de processos e fluxos de informações e materiais que podem ocorrer entre as mesmas. Um processo é uma coleção de atividades, as quais assumem um número de entradas e produzem um número de saídas. Informações podem ser passadas de uma atividade para a outra contendo conceitos importantes no processo organizacional sendo representado. Um material representa um conjunto de elementos do mundo real que podem ser transportados de uma atividade para outra. A estrutura do modelo de atividades é similar ao Modelo DFD da análise estruturada, por conter fluxos de informações e materiais que entram e saem das atividades.
5. **modelo de requisitos do sistema de informação (MRSI):** Este modelo permite declarar e analisar os objetivos, problemas e requisitos do sistema de informação que está sendo projetado. É utilizado como base para a elaboração de uma especificação de requisitos acordada e claramente compreendida por todos os *stakeholders*. O modelo apresenta algumas nomenclaturas com semânticas bem definidas para representar aspectos dos requisitos dos sistemas de informação. A expressão *É Objetivo*; é utilizada para expressar intenções relacionadas com o sistema de informação. *É*

Problema; expressa estados ou fatos não desejados. *É Limitação*; é utilizada para determinar algum tipo de restrição em relação ao sistema de informação ou em relação ao processo de construção deste sistema. *É requisito*; especifica um requisito do sistema, podendo ser funcional ou não-funcional. *Característica Valiosa*; expressa uma característica importante em relação ao sistema de informação que não foi definida como um requisito, mas sim, como propriedade desejável do sistema. Esta propriedade pode, eventualmente, não ser obtida devido à indisponibilidade de tempo ou de outros recursos necessários.

Os modelos apresentados acima são todos inter-relacionados para compor o modelo organizacional completo proposto por Bubenko (1993). Cada elemento no Modelo de Objetivos (MO), por exemplo, motiva o aparecimento de elementos em cada um dos outros modelos. Regras no Modelo de Objetivos (MO), podem controlar entradas de Material e Informação em processos definidos no Modelo de Atividades e Uso (MAU). Atores no Modelo de Atores (MA), podem ser responsáveis por *Objetivos* ou podem originar nós no Modelo de Objetivos (MO), bem como nos outros modelos propostos na técnica. Atores podem também ser realizadores, iniciadores ou responsáveis por atividades nos processos definidos no Modelo de Ações e Uso (MAU).

Em Bubenko (1993), são apresentados de uma forma mais especifica, os principais objetivos do uso da técnica de modelagem organizacional no contexto da Engenharia de Requisitos. São eles:

- melhorar e documentar o conhecimento dos participantes em relação à situação atual da organização e situações futuras desejáveis. Estas situações, tipicamente podem incluir o desenvolvimento de novos sistemas de informação;

- melhorar as possibilidades de se obter de forma clara, estruturada e documentada, a concordância dos envolvidos a respeito de conceitos importantes, propriedades, atividades e objetivos desejáveis em relação a situações futuras da organização;
- desenvolver uma base, tão completa quanto possível, para o projeto de sistemas de informação que satisfaça adequadamente os objetivos organizacionais;
- desenvolver um conjunto de documentos inter-relacionados que no futuro possam ser reutilizados para re-projetar objetivos, atividades e requisitos de sistemas de informação.

Da mesma forma que na técnica i*, a técnica de Bubenko, não propõe heurísticas que permitam relacionar de forma sistemática, requisitos organizacionais a requisitos funcionais e não funcionais de sistemas de software. Há a necessidade evidente de se estabelecer mecanismos que permitam que engenheiros de requisitos possam utilizar de forma mais efetiva as informações apresentadas nos modelos de Bubenko.

No entanto, para efeitos de aplicabilidade na nossa proposta, consideramos que a técnica i* é de mais fácil entendimento, necessita de menos esforço para ser utilizada, bem como tem um bom poder de expressividade dos elementos do ambiente organizacional, sendo mais apropriada para ser integrada com técnicas baseadas em cenários. Além disso, pesquisas recentes têm adotado i* como ferramenta importante no desenvolvimento de software (Castro et al. 2002) (Mylopoulos et al. 2000) (Mylopulos et al. 1999).

4.3 Modelagem Organizacional no RUP

O RUP ("*Rational Unified Process*") é um processo de desenvolvimento de software bastante utilizado na área comercial e acadêmica (Kruchten 2000). Basicamente, apresenta as seguintes características:

- trata-se de um processo orientado a objetos;
- utiliza-se princípios modernos de Engenharia de software tais como: desenvolvimento iterativo e incremental, arquitetura de software e outros;
- preocupa-se com a problemática envolvendo a execução de processos de software;
- trata-se de um processo criado por autores (Jacobson 1995) (Booch 1992) (Rumbaugh 1994), bastante familiarizados com práticas de desenvolvimento de software tanto na área acadêmica como industrial;

O processo unificado, através do estabelecimento de alguns princípios, visa o desenvolvimento de software de qualidade. Estes princípios podem ser assim descritos:

1. **o processo unificado adota casos de uso como elemento fundamental no desenvolvimento de software**: No processo unificado, casos de uso são centrais no processo de desenvolvimento.
2. **o processo unificado é centrado em arquitetura de software**: o processo unificado fundamenta-se também em uma arquitetura que define inicialmente os principais elementos do software, representados principalmente pelos casos de uso mais importantes, subsistemas, classes e componentes, bem como pelas colaborações entre estes elementos através de interfaces.

3. **o processo unificado é iterativo e incremental:** o processo unificado subdivide o desenvolvimento de software em quatro grandes **fases** denominadas de **Concepção, Elaboração, Construção e Transição**, sendo que cada fase possui alguns objetivos e conteúdos associados, como segue:

    - ***concepção***: nesta fase objetiva-se definir os casos de uso mais críticos, os quais representam as funções chaves do sistema. Delimita-se o escopo do produto a ser desenvolvido, identifica-se e reduz-se principalmente os riscos críticos.
    - ***elaboração***: nesta fase define-se a descrição arquitetural do software. Procura-se também definir a maioria dos casos de uso, capturando a maioria dos requisitos do software. No final desta fase, deve-se estar apto a planejar a fase de construção em detalhes.
    - ***construção***: o software deve ser construído completamente. Ou seja, deve-se adicionar a musculatura ao esqueleto (arquitetura). Visa-se a capacidade operacional do software.
    - ***transição***: esta fase envolve a realização de testes com o usuário, corrigir defeitos encontrados e realizar treinamento.

    Cada uma destas fases é composta de cinco ***workflows*** denominados de **Modelagem de Negócio, Requisitos, Análise, Projeto, Implementação e Teste**. Dependendo da fase, maior ou menor nível de detalhes podem ser desenvolvidos em um *workflow* específico.

    O conceito de processo **iterativo** vem da propriedade de que cada fase pode conter uma ou mais iterações de desenvolvimento, sendo que cada iteração realiza um ciclo completo nos *workflows*. Isto significa dizer

que, cada iteração pode ser encarada como a execução do modelo de Cascata. Depois de concluída uma iteração, ocorre uma evolução, a qual **incrementa** funcionalidade e/ou melhorias.

Voltando o foco para o aspecto da **modelagem organizacional**, podemos observar que o Processo Unificado da Rational define o *workflow* **Modelagem de Negócio**, o qual tem como objetivo modelar a organização. Os principais propósitos da modelagem de negócio são:

- compreender a estrutura e as dinâmicas da organização na qual o sistema será desenvolvido e/ou entregue;
- compreender os problemas atuais na organização e identificar melhorias potenciais;
- garantir que clientes, usuários e desenvolvedores tenham uma compreensão comum da organização;
- derivar os requisitos do sistema necessários para suportar a organização.

Para obter estes objetivos, o *workflow* de modelagem de negócio descreve como desenvolver uma visão sobre a organização alvo, e com base nesta visão define os processos, regras e responsabilidades da organização em um Modelo de Casos de Uso de Negócio e em um Modelo de Objetos de Negócio. Estes dois modelos podem ser assim definidos:

- ***modelo de casos de uso de negócio***: modelo que representa as funções de negócio desejadas. Este modelo é utilizado como uma entrada essencial para identificar papéis e responsabilidades na organização. Um modelo de casos de uso de negócio consiste de atores e casos de uso a nível de

negócio. Os atores representam papéis externos em relação a um negócio (por exemplo, clientes), e casos de uso de negócio são processos.

- ***modelo de objetos de negócio***: um modelo de objetos de negócio inclui realizações de casos de uso, as quais mostram como casos de uso de negócio são "realizados", em termos de interações de trabalhadores de negócio e entidades de negócio.

Assim, basicamente estes são os recursos utilizados pelo RUP ("*Rational Unified Process"*), que permitem a modelagem da organização. Neste contexto, e observando as outras técnicas de modelagem organizacional, tais como: i* e Bubenko apresentadas nas seções anteriores, consideramos que os dois modelos propostos no RUP ("*Rational Unified Process"*) para modelagem de negócio, têm menor poder de expressividade e não permitem modelar uma série de questões importantes na organização, tais como: motivações, intenções e razões que estão associadas aos processos organizacionais. A captura deste conhecimento enriquece modelos organizacionais, mostrando a complexidade dos relacionamentos da organização. Técnicas tais como as propostas no RUP ("*Rational Unified Process"*), são mais apropriadas para descrever os "o quês" e não permitem expressar os "porquês" adequadamente.

Por outro lado, observando o aspecto que está relacionado à possibilidade de evolução de modelos de negócio para modelos de sistemas computacionais, verificamos que uma das vantagens apontadas pelos criadores do RUP ("*Rational Unified Process"*) é que este processo permite mostrar de forma mais clara e concisa, as dependências entre negócios e sistemas. Este relacionamento é mostrado na figura 16.

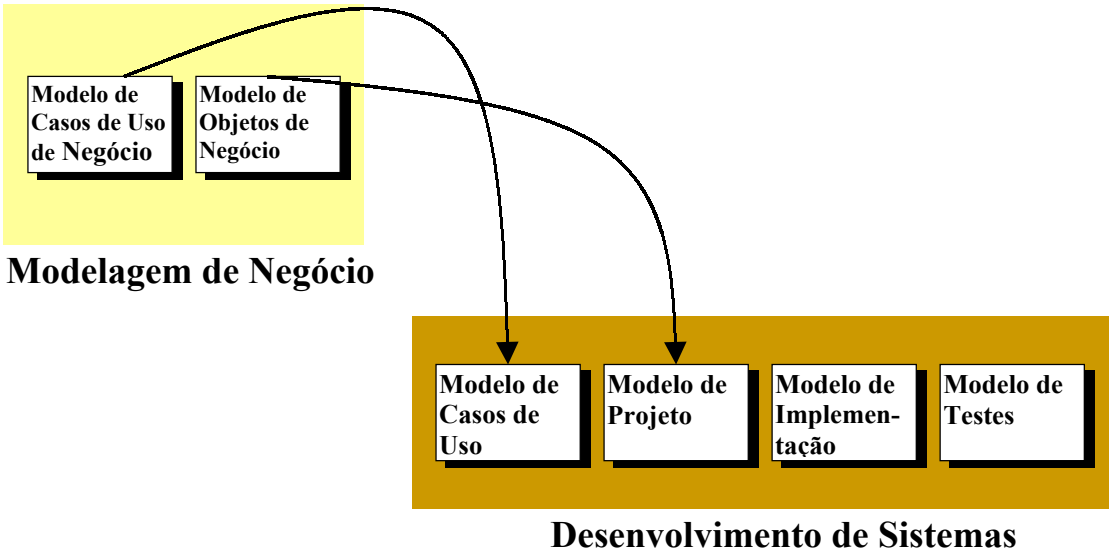

Figura 16. Relacionamentos entre modelos de negócio e modelos de sistemas no RUP.

Contudo, consideramos que o processo de derivação de modelagem de negócio para modelagem de sistemas ainda exige uma grande quantidade de análise, inclusive por desenvolvedores mais experientes. O processo de construção, tanto de modelos de casos de uso e objetos de negócio quanto de casos de uso e objetos do sistema, não é trivial e as diretrizes apresentadas no RUP ("*Rational Unified Process"*) ainda são insuficientes para tornar o trabalho como um todo sistemático. Tipicamente, ainda permanecem os problemas com a elicitação e especificação de casos de uso e objetos. Os conceitos associados com estas técnicas, em muitos casos, não são bem compreendidos pelos *stakeholders*, o que pode dificultar o trabalho, bem como levar a construção de modelos inconsistentes e/ou incompletos. Além disso, consideramos que a não captura dos "porquês" por estas técnicas, também torna os modelos de sistemas derivados menos expressivos, podendo inclusive não incluir aspectos importantes no contexto do uso de sistemas computacionais. Destacamos que nesta tese, modelos organizacionais em i* são utilizados para derivar requisitos de sistemas computacionais, mas também e não menos importante, consideramos que estes modelos são ferramentas

essenciais para documentar e poder acompanhar/rastrear a evolução dos processos de negócio e sistemas computacionais que apóiam estes processos.

4.4 Considerações Finais

Neste capítulo, apresentamos a descrição de algumas técnicas de modelagem organizacional bastante conhecidas na Engenharia de Requisitos: i* e Bubenko e a perspectiva adotada no RUP ("*Rational Unified Process"*). Estas técnicas apresentam diferentes modelos para representar o conhecimento sobre processos organizacionais/negócio e sobre os relacionamentos entre os membros de uma organização. A representação destas informações é importante para compreender adequadamente a organização e poder elicitar com mais efetividade e consistência os requisitos de sistemas computacionais pretendidos.

A partir do conhecimento do ambiente organizacional, engenheiros de requisitos podem definir como o sistema de software pretendido irá satisfazer os objetivos da organização, por que ele é necessário, quais as alternativas existentes, quais as implicações das alternativas para as várias partes interessadas, entre outros aspectos. Além disso, uma série de informações representadas em modelos organizacionais podem ser utilizadas e mapeadas para requisitos funcionais e não funcionais do sistema pretendido.

Consideramos que a técnica de i* estabelece relacionamentos entre atores (modelo de dependências estratégicas), bem como mecanismos para representar as razões associadas com as intenções de atores (modelo de razões estratégicas), que podem servir para dar origem à descrição de casos de uso e cenários para sistemas de software pretendidos. As dependências em i* do tipo objetivo, tarefa, recurso e objetivo-soft, bem como as ligações do tipo meio-fim e de decomposição de tarefa,

utilizadas para decompor em mais detalhes as razões associadas com cada ator e suas dependências, constituem informações bastante úteis que podem derivar requisitos funcionais e não funcionais de sistemas. O conhecimento e compreensão destas informações pelos diversos *stakeholders*, pode também facilitar o trabalho de obter concordância e estabelecer um compromisso entre todas as partes envolvidas em relação à definição dos requisitos de sistemas computacionais.

No próximo capítulo, com base nos estudos realizados sobre técnicas de modelagem organizacional e técnicas baseadas em cenários integradas com abordagens orientadas a objetivos, apresentamos a nossa proposta de integração de i* e Casos de Uso em UML ("*Unified Modeling Language*"). Descrevemos algumas diretrizes para auxiliar nesta integração e posteriormente aplicamos estas diretrizes a dois estudos de caso.

Capítulo 5

Integrando Modelagem Organizacional com Casos de Uso

Apresentamos nos capítulos anteriores as principais características relacionadas com algumas técnicas baseadas em cenários, enfatizando a abordagem de casos de uso, e com técnicas utilizadas para a modelagem organizacional, focando na técnica i*. Estas descrições objetivaram fundamentar a nossa proposta de integração de modelos organizacionais com casos de uso. Tanto técnicas de modelagem organizacional quanto casos de uso não englobam/lidam satisfatoriamente com as várias características inerentes ao ambiente organizacional, bem como aos sistemas computacionais pretendidos.

Modelos organizacionais não detalham explicitamente requisitos funcionais, mas podem fornecer informações sobre o ambiente organizacional que podem auxiliar nesta tarefa, entre as quais destacamos: os objetivos gerais da organização, intenções dos atores da organização em relação a sistemas pretendidos, detalhamento das razões associadas com atores para atingir determinados objetivos e descrição de requisitos não funcionais. Estas informações servem para compreender melhor os "porquês" associados com processos organizacionais e com sistemas de softwares que auxiliam na execução destes processos. Normalmente, o que ocorre na Engenharia de Requisitos é utilizarmos modelos que representam "o que" nós queremos dos sistemas de software e "como" estes sistemas devem ser estruturados, por exemplo, através de DFDs (Diagrama de Fluxo de Dados), DER (Diagrama de Entidade Relacionamento), Diagramas em UML ("U*nified Modeling Language"*), etc. Contudo, é importante conhecer as razões e motivações associadas ao que esperamos de sistemas

computacionais. Isto significa explorar elementos de processos que permitam representar todos os aspectos relevantes (razões) que fundamentam "o que" pretendemos. A Engenharia de Requisitos orientada a objetivos (*GORE – Goal Oriented Requirements Engineering*) tem tido um papel fundamental para apontar estratégias e técnicas que permitam representar melhor estes "porquês" (ver capítulo 3). Por sua vez, técnicas baseadas em cenários, em geral e mais especificamente casos de uso, são utilizados para descrever requisitos funcionais, não lidando adequadamente com aspectos relacionados a objetivos e requisitos organizacionais. Assim, os casos de uso não lidam adequadamente com as razões ("porquês"), que estão por detrás de sistemas computacionais pretendidos. Uma melhor compreensão destas razões pode ser obtida através da descrição do ambiente organizacional usando a técnica i*. Possibilita-se assim investigar dependências estratégicas entre atores, bem como razões estratégicas destes atores. i* permite que possamos, inclusive, modelar as expectativas e objetivos de atores da organização em relação a sistemas computacionais pretendidos, possibilitando também avaliar o impacto que a introdução destes sistemas de software podem provocar nos processos de negócio e conseqüentemente nas atividades diárias dos atores da organização.

Sob esta visão, as expectativas de atores da organização sobre sistemas computacionais pretendidos, podem ser traduzidas/mapeadas em objetivos de casos de uso desses sistemas computacionais (seguindo algumas heurísticas que serão apresentadas na nossa proposta). Por outro lado, as razões e expectativas não associadas diretamente com sistemas pretendidos podem dar origem também a casos de uso de negócio. Desta forma, utilizar modelos organizacionais como fonte de informações para a descrição de casos de uso, permite que engenheiros de requisitos possam estabelecer um relacionamento entre os requisitos funcionais e os objetivos previamente definidos

na organização. É importante também ressaltar que todo o nosso processo de mapeamento tem como base o estudo de técnicas originadas da *GORE* (*Goal Oriented Requirements Engineering*). Isto porque, cada elemento em i*, é observado como potencialmente apto a ser mapeado para um Objetivo (Goal) de um caso de uso do sistema ou de negócio. Isto implica em realizar uma série de questionamentos, os quais tipicamente são perguntas que tentam extrair dos modeladores de i*, os objetivos, ou seja, o que de fato se deseja alcançar quando se estabelece uma dependência entre os atores organizacionais. Estes questionamentos assemelham-se aos questionamentos aplicados na técnica "*Inquire Cycle*" (Potts 1999) (Anton 1997) para descobrir requisitos de sistemas. Assim, através de uma análise orientada a objetivos dos modelos organizacionais, pode-se derivar/mapear objetivos, intenções e motivações de atores da organização para objetivos principais de casos de uso. Assumimos, que para cada caso de uso representando interações entre um usuário e o sistema, associamos um objetivo principal, que representa o que o usuário espera obter de valor como resultado da execução do caso de uso (veja capítulo 3).

Desta forma, observamos que o fato de propor mecanismos que permitam integrar ambas as técnicas pode tornar o processo de elicitação, análise e documentação de requisitos mais sistemático, pois informações sobre o ambiente organizacional antes não consideradas, passam na nossa proposta, a nortear a realização das atividades do processo de Engenharia de Requisitos. A partir desta integração, organizações e engenheiros de requisitos podem guiar e conduzir o processo de Engenharia de Requisitos com base nos objetivos e processos organizacionais previamente estabelecidos. A descrição de casos de uso, com base nestes elementos, pode então iniciar de um ponto (modelos organizacionais), bem conhecido e compreendido por

todos os *stakeholders*. Isto pode evitar futuras inconsistências, ambigüidades e incertezas a respeito dos requisitos de sistemas pretendidos.

Neste capítulo, apresentamos a nossa proposta de integração de casos de uso e modelos organizacionais (i*). Inicialmente na seção 5.1, apresentamos algumas considerações importantes a respeito do processo de integração proposto. Na seção 5.2, apresentamos a proposta de integração da técnica i* com a técnica de Caso de Uso. São descritas as diretrizes propostas para auxiliar engenheiros de requisitos a descrever Casos de Uso, a partir dos modelos de Dependências Estratégicas (SD) e Razões Estratégicas (SR) desenvolvidos com a técnica i*. Paralelamente à descrição da proposta, apresentaremos exemplos da aplicação de cada diretriz, tomando como base o exemplo do problema clássico de Agendamento de Reuniões (figuras 13 e 14 no capítulo 4) e o exemplo de uma aplicação de comércio eletrônico (figuras 18 e 19 no capítulo 6). Na seção 5.3, apresentamos as considerações finais do capítulo.

5.1 Uma visão geral do processo de integração de modelos organizacionais com casos de uso

No primeiro capítulo de nosso trabalho, apresentamos na figura 1, os elementos básicos da nossa proposta. Basicamente, objetivamos integrar os modelos organizacionais representados através da técnica i* com casos de uso. A primeira pergunta que surge é como é possível integrar duas técnicas com enfoques diferentes. i* visa, basicamente, representar o ambiente organizacional e provê suporte para a fase inicial ("*early requirements*") da Engenharia de Requisitos. Já casos de uso têm seu enfoque na descrição dos requisitos funcionais de sistemas pretendidos ou processos de negócio (isto é, casos de uso de negócio). A questão chave é que i* permite representar/modelar aspectos do ambiente organizacional considerados essenciais para apoiar a fase inicial de Engenharia de Requisitos (Yu 1995), propiciando principalmente

representar as razões estratégicas ("porquês") por detrás de processos na organização. Além disso, existem razões estratégicas que estão associadas com a necessidade de desenvolver sistemas computacionais para apoiar atividades na organização. Estas razões são uma fonte de informação para a definição e descrição de requisitos para estes sistemas computacionais. Por outro lado, relacionamentos estratégicos entre atores que não englobam o sistema computacional, podem dar origem a casos de uso de negócio. Sob esta visão, surge uma forma de poder estabelecer os limites entre o que pertence ao escopo de sistemas computacionais e o que pertence ao escopo de processos de negócio.

No entanto, surge uma pergunta neste contexto: como podemos tornar viável o mapeamento entre elementos de i* e casos de uso? É necessário definir heurísticas que possam nos auxiliar no estabelecimento desta correlação. Salientamos que não se encontra na literatura da área hoje, quaisquer heurísticas que permitam derivar casos de uso a partir de i*. Com base na análise de ambas as técnicas, bem como no uso das heurísticas definidas nesta tese, em alguns estudos de caso (ver capítulo 6), temos chegado a um conjunto estável de diretrizes que permitem derivar casos de uso mais estáveis e melhor compreendidos pelos *stakeholders*. Isto decorre da própria integração dos stakeholders tanto no processo de construção dos modelos em i* quanto do processo mais consistente de derivação de casos de uso a partir de modelos organizacionais.

Neste ponto, é importante também lembrar que a nossa investigação de elementos em i*, visando derivar casos de uso, é fundamentada em uma análise orientada a objetivos fruto do estudo de vários trabalhos na conhecida Engenharia de Requisitos Orientada a Objetivos (*GORE – Goal Oriented Requirements Engineering*) (Potts 1997) (Rolland et al. 1998) (Anton 1997) (Yu 1995) (Yu et al. 1997) (Castro et al. 2002) (Kaiva et al. 2002) (Mylopoulos et al. 1999).

Para entender a análise orientada a objetivos adotada nesta tese, relembremos alguns conceitos apresentados na seção 3.4, na qual fizemos uma breve descrição de uma abordagem de construção de cenários, que utiliza um tipo de modelagem de objetivos para capturar requisitos. No ambiente descrito, denominado de L'Ecritoire (Rolland et al. 1998), argumenta-se que a análise orientada a objetivos pode ser útil na construção de cenários (ver seção 3.2). No entanto, em L'Ecritoire, a descoberta dos objetivos iniciais de usuários/organização, que podem levar a definição e descrição de cenários, depende em grande parte, da capacidade de engenheiros de requisitos em elicitar estes objetivos. Não são apresentadas heurísticas que possam auxiliar engenheiros de requisitos neste trabalho, nem modelos que possam servir de fonte de informação para a descoberta de objetivos. Corre-se também o risco, de que usuários não estejam cientes ou não conheçam adequadamente quais são os objetivos estratégicos da organização, bem como quais são as intenções reais em relação aos sistemas pretendidos. Esta estratégia é bastante interessante, mas insatisfatória, dado que engenheiros de requisitos normalmente não possuem informações explícitas e modeladas a respeito dos objetivos organizacionais e/ou das intenções dos atores da organização em relação a sistemas pretendidos.

Este tipo de análise reforça a nossa idéia, pois observando i*, verificamos que são definidas uma série de objetivos (*goals*), os quais podem diretamente servir de fonte de informação para a descoberta de novos cenários. Além disso, existem outras dependências entre atores em i*, do tipo objetivo-soft, tarefa e recurso, que são relacionadas diretamente a sistemas computacionais pretendidos. Podemos analisar estas dependências, procurando associá-las a objetivos sobre sistemas computacionais que possam ser mapeados para objetivos de casos de uso e cenários do sistema a ser desenvolvido. Desta forma, salientamos que nesta tese, objetivos são os elementos a

partir dos quais podemos descobrir e refinar os requisitos funcionais para sistemas computacionais pretendidos. Estes requisitos funcionais são representados e descritos na nossa abordagem através de casos de uso e cenários associados.

Nesta tese, na qual visamos mostrar a viabilidade de integrar a técnica ***i**** com ***Casos de Uso***, definimos algumas diretrizes, visando analisar as dependências entre os atores organizacionais do ponto de vista dos objetivos que podem e/ou desejam ser obtidos pelos mesmos. Estas diretrizes, bem como um rol maior de diretrizes definidas para integrar de forma mais eficiente os elementos descritos em i* com casos de uso, serão apresentadas na próxima seção. De uma forma geral, as diretrizes que propomos visam permitir que a partir das dependências presentes nos modelos de Dependências Estratégicas (SD) em i*, do tipo objetivo, objetivo-soft, recurso e tarefa, engenheiros de requisitos possam descobrir casos de uso para um sistema de software da organização. Descrevemos também diretrizes, visando a análise das informações (ligações de decomposição de tarefa e ligações meio-fim) definidas no modelo de Razões Estratégicas, para *especificar* os fluxos principal e alternativos (cenários primário e secundários) dos casos de uso descobertos para o sistema.

Apresentaremos o processo de integração de i* e casos de uso, bem como as diretrizes propostas na próxima seção. Antes de descrever em mais detalhes a nossa proposta, precisamos fazer algumas considerações, as quais são necessárias para um melhor entendimento dos nossos objetivos:

1. neste trabalho, propomos a integração da técnica i* com cenários sob a forma de Casos de Uso. No capítulo 7, no qual descrevemos trabalhos futuros, expomos também a possibilidade de aplicar nossa proposta para outras técnicas baseadas em cenários tais como Leite et al. (1997) e Crews (Ralyté 1999) (Rolland et al. 1998);

2. adotamos como base para a nossa proposta, o projeto *Tropos* (Castro et al. 2000) (Mylopoulos et al. 2000) (Castro et al. 2001) (Castro et al. 2002). Este projeto propõe fundamentalmente o desenvolvimento de software orientado a requisitos, o qual inicia-se com o desenvolvimento de modelos organizacionais através da técnica i*. Assim, adotamos nesta tese que os primeiros elementos a serem produzidos no processo de Engenharia de Requisitos são os Modelos de Dependência (SD) e de Razões Estratégicas (SR) desenvolvidos com a técnica i*. Estes modelos constituem a base para a realização das outras atividades do processo de Engenharia de Requisitos. Preferencialmente, devem ser incluídos nestes modelos, descrições sobre sistemas computacionais que possam auxiliar nas tarefas diárias da organização;
3. adotamos como base para o processo de integração de ambas as técnicas i* e Casos de Uso, os conceitos da Engenharia de Requisitos Orientada a Objetivos (*GORE – Goal Oriented Requirements Engineering*) (Anton 1997) (Potts 1997) (Rolland et al.1998) visando extrair objetivos (os quais podem ser associados a casos de uso) a partir das informações definidas nos modelos organizacionais. O processo de desenvolvimento de Casos de Uso é realizado, procurando-se definir objetivos que possam ser derivados a partir dos relacionamentos de dependências do tipo objetivo, recurso, tarefa e objetivo-soft, definidos nos modelos de Dependências Estratégicas (SD). Cada objetivo definido em um nível adequado de abstração originará um caso de uso e será o principal propósito/objetivo do caso de uso. Adotamos que cada caso de uso possui um objetivo principal que representa o que um usuário espera obter com a sua execução (Cockburn 2000).

Os passos dos fluxos principal e alternativos (ver seção 3.2) de casos de uso podem também ser extraídos a partir dos elementos descritos no Modelo de Razões Estratégicas em i*. As diretrizes que propomos para este fim serão descritas na próxima seção;

4. traduziremos os objetivos-soft descritos como dependências em i* (nos Modelos de Dependências Estratégicas) entre atores organizacionais, mapeados para atores do sistema; e o ator que representa o sistema computacional pretendido, em requisitos não-funcionais do sistema computacional pretendido. Faremos também uma análise dos objetivos-soft internos à decomposição dos atores nos Modelos de Razões Estratégicas (SR) buscando associar estes objetivos-soft com requisitos não funcionais dos casos de uso descobertos para o sistema pretendido. No entanto, consideramos esta visão de descrição de requisitos não funcionais ainda deficiente, pois não trata entre outros aspectos, interdependências entre requisitos não funcionais, bem como a necessária decomposição de requisitos mostrando sua abrangência e impacto em sistemas de software. Futuros trabalhos lidarão com esta questão;
5. tomaremos como base, na nossa proposta, para o desenvolvimento de um documento de requisitos mais completo e consistente, tanto os modelos organizacionais quanto os casos de uso. A maioria dos processos de desenvolvimento de software orientados a objetos existentes na literatura, tais como: Rational Unified Process (Jacobson et al. 1999) e Catalysis (D'Souza et al. 1998), adotam casos de uso como base para o desenvolvimento, não tratando satisfatoriamente, no entanto, os objetivos e metas organizacionais que impulsionam e conduzem o desenvolvimento de

novos sistemas de software. Isto dificulta um maior envolvimento e comprometimento dos *stakeholders* em relação aos requisitos do sistema e torna o processo de descoberta e descrição de casos de uso *ad-hoc* e insatisfatório, ainda que se utilizando das heurísticas apresentadas na literatura (Schneider et al. 1998). Recentemente, algumas pesquisas têm tratado de mostrar como UML pode mais efetivamente ser usado para modelagem de negócio (Eriksson et al. 2000). No entanto, estas pesquisas ainda estão em andamento.

5.2 Integrando i* com Casos de Uso

Nossa proposta parte do princípio que uma organização utiliza modelos organizacionais desenvolvidos através da técnica i* como base para o entendimento e comunicação dos envolvidos no ambiente organizacional. Tanto o Modelo de Dependências Estratégicas quanto o Modelo de Razões Estratégicas da técnica i* (Yu 1995), devem ser desenvolvidos incluindo possíveis alternativas de sistemas computacionais que possam auxiliar a organização na obtenção de suas metas.

A partir dos modelos organizacionais SD e SR iniciamos o processo de integração e descoberta dos casos de uso para um sistema alvo da organização. Inicialmente, são descobertos os atores para os diagramas de Caso de Uso em UML e posteriormente, são descobertos os casos de uso para estes atores, bem como os fluxos principal e alternativos (cenários primário e secundários) dos casos de uso descobertos. Na figura 17, apresentamos uma visão geral do processo de integração de i* e Casos de Uso. Nesta figura, os passos 1, 2 ,3 (retângulo de cantos arredondados) representam, respectivamente, as atividades de descoberta de atores, de descoberta de casos de uso para os atores e de descrição/especificação dos fluxos de interações (cenários) para os

casos de uso. A entrada para o processo de integração é composta pelos Modelos de Dependências e Razões Estratégicas em i*. Nos passos 1 e 2, a entrada é o Modelo de Dependências Estratégicas. A descrição dos fluxos principal e alternativos (passo 3) dos casos de uso é derivada do Modelo de Razões Estratégicas. Os resultados do processo de integração são os diagramas de casos de uso para o sistema computacional, bem como descrições textuais dos fluxos principal e alternativos dos casos de uso. Para cada um dos passos do processo de integração, descrevemos diretrizes para auxiliar engenheiros de requisitos na realização do passo. Estas diretrizes conduzem o engenheiro de requisitos na realização de uma análise orientada a objetivos dos modelos organizacionais SD e SR, procurando derivar objetivos que potencialmente podem originar casos de uso.

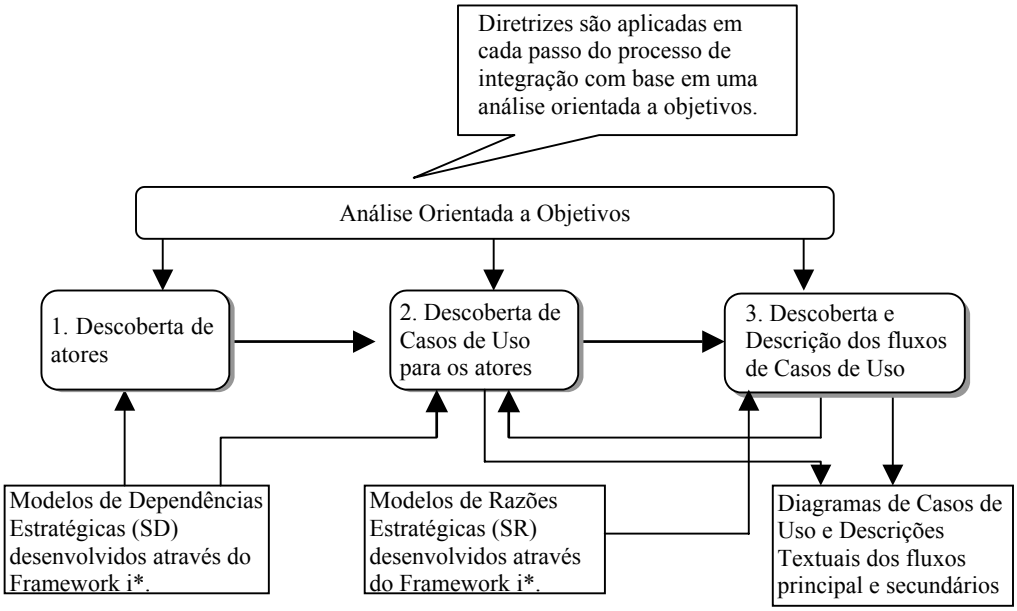

Figura 17. Uma visão geral do processo de integração de i* e Casos de Uso.

Podemos também observar na figura 17, que os passos 2 (descoberta de casos de uso para os atores) e 3 (descoberta e descrição dos fluxos para os casos de uso), podem ser realizados de forma intercalada/iterativa, sendo que passos nas descrições de casos

de uso (cenários), podem originar novos casos de uso, que por sua vez, terão associados novos objetivos. Apresentaremos, posteriormente, algumas heurísticas originadas da técnica GBRAM (Anton 1997), que poderão auxiliar engenheiros de requisitos na decisão se determinado passo em descrições de casos de uso, necessita ser refinado e descrito em um novo caso de uso. De forma geral, na prática, o processo de integração de i* e casos de uso é bastante iterativo, sendo que a cada iteração, novos casos de uso e especificações/descrições para estes casos de uso podem ser descobertas.

Na seqüência, descrevemos as diretrizes propostas para derivar casos de uso em UML a partir de i*. Versões preliminares da nossa proposta foram descritas em (Santander et al 2002a) (Santander et al 2002) (Santander et al. 2001). Estas diretrizes estão relacionadas a propósitos específicos, conforme segue:

1º Passo da Proposta: Descoberta de 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:

SubDiretriz 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>>.

2º Passo da Proposta: 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;

SubDiretriz 5.1: deve-se avaliar as dependências do tipo objetivo associadas com o ator;

SubDiretriz 5.2: deve-se avaliar as dependências do tipo tarefa, associadas com o ator;

SubDiretriz 5.3: deve-se avaliar as dependências do tipo recurso, associadas com o ator;

SubDiretriz 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).

3º Passo da Proposta: Descoberta e descrição do fluxo 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.

A seguir, nas seções 5.2.1, 5.2.2 e 5.2.3 descrevemos em detalhes, respectivamente, as diretrizes para os passos 1 , 2 e 3 da nossa proposta.

5.2.1 Diretrizes para a descoberta de atores em casos de uso a partir do framework i*

Conforme descrito na seção 3.1, normalmente o primeiro passo na construção de Diagramas de Casos de Uso é a descoberta dos atores do sistema. Neste sentido, detectamos que o **ator** (atores) é um elemento importante e fundamental em ambas as técnicas i* e Casos de Uso. No entanto, existem algumas diferenças nos conceitos associados a um ator nas duas técnicas. É importante observar estas diferenças no momento do mapeamento de atores em i* para atores em casos de uso. A tabela 1 apresenta de forma resumida, alguns conceitos relacionados a atores em ambas as técnicas.

Tendo em vista algumas diferenças de conceitos de ator nas técnicas estudadas (ver tabela 1), é necessário estabelecer algumas diretrizes para auxiliar no processo de mapeamento/relacionamento de atores em i* com Casos de Uso. A seguir, apresentamos as diretrizes que permitem identificar candidatos a atores em Casos de Uso a partir dos atores definidos em i*. Estas diretrizes constituem o primeiro passo da nossa proposta.

Características de um Ator nas Técnicas i* e Casos de Uso	
Ator em i*	***Ator em Caso de Uso***
Um ator é qualquer entidade que possua intenções em um ambiente organizacional. Define-se dependências de vários tipos entre atores, para representar "como" e "porque" atores dependem uns dos outros no ambiente organizacional. Tipicamente, um ator em i* espera obter algo relevante de sistemas computacionais pretendidos.	Um ator pode incluir pessoas, sistemas ou máquinas que interagem com o sistema a ser desenvolvido;
Pode-se modelar um ator "genérico" ou "social" tendo relacionamentos estratégicos com outros atores.	Um ator desempenha um papel específico em relação ao sistema.
Atores podem representar partes/componentes do sistema ou o próprio sistema computacional.	Atores são sempre partes externas ao sistema. Eles nunca são partes componentes (internas) do sistema.
Objetiva-se modelar as dependências estratégicas entre atores bem como as razões estratégicas internas de cada ator que motivam suas ações.	Objetiva-se descrever as ações de um ator interagindo com o sistema.

Tabela 1. Características associadas com um Ator nas técnicas i* e Caso de Uso.

Visando facilitar a leitura das diretrizes, estamos repetindo a seguir o Modelo de Dependências Estratégicas (figura 18) e o Modelo de Razões Estratégicas (figura 19), do exemplo de agendamento de reuniões apresentado no capítulo 4.

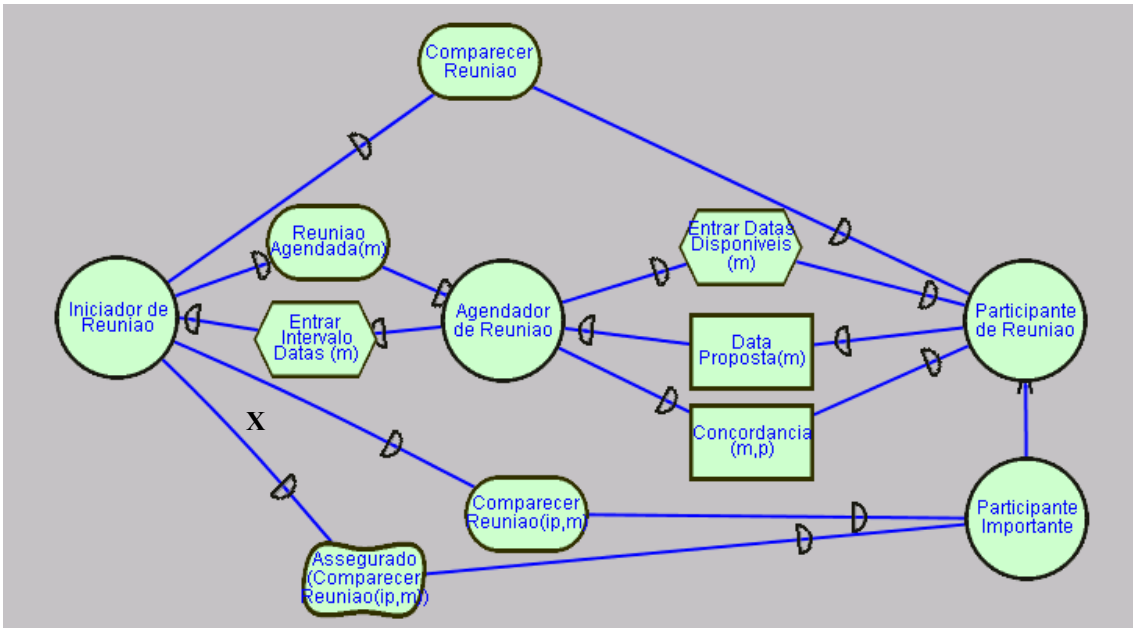

Figura 18. Modelo de Dependências Estratégicas para agendamento de reuniões.

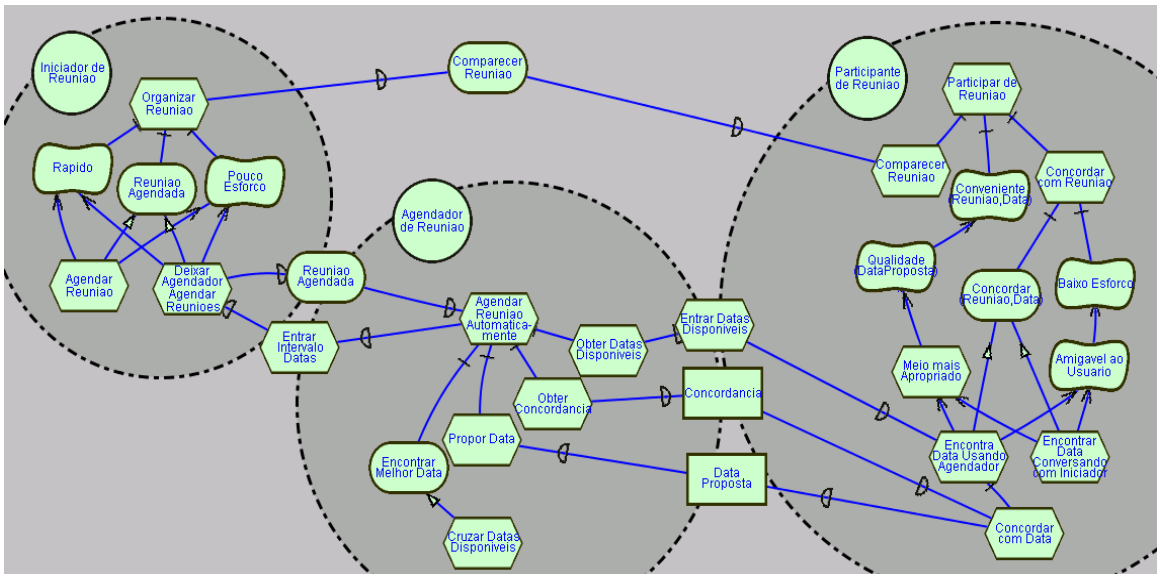

Figura 19. Modelo de Razões Estratégicas para agendamento de reuniões.

1º Passo: Descoberta de Atores

Diretriz 1: todo ator, no modelo em i*, deve ser analisado em um possível mapeamento para ator em Casos de Uso;

> *Para um melhor entendimento das diretrizes, analisemos, por exemplo, o ator Participante de Reunião, apresentado no Modelo de Dependências Estratégicas da figura 18;*

Diretriz 2: inicialmente, deve-se verificar se o ator escolhido é **externo** ao sistema computacional. Atores em casos de uso nunca são partes do sistema. É necessário observar este aspecto, pois na modelagem organizacional pode-se incluir atores representando partes do sistema ou até mesmo uma abstração do software como um todo, o que na técnica de Casos de Uso não caracteriza um ator;

> *Observando a figura 18, verificamos que o ator Participante de Reunião participa do processo de agendamento de reuniões que considera a existência de um sistema computacional denominado Agendador de Reunião para auxiliar neste processo. Claramente o ator Participante de Reunião pode ser considerado externo ao sistema, já que o mesmo interagirá com o sistema computacional a ser desenvolvido, fornecendo informações importantes para o processo de agendamento. Sob este prisma, o ator em questão é um candidato potencial a ator em Casos de Uso.*

Diretriz 3: se o ator sendo avaliado é considerado externo ao sistema computacional, conforme passo anterior, é necessário garantir o seu mapeamento para ator no Diagrama de Casos de Uso, através da seguinte análise:

Diretriz 3.1: deve-se garantir que o ator interagirá com o sistema pretendido - isto pode ser observado, se existir pelo menos uma dependência entre o ator sendo analisado e o ator que representa o sistema computacional pretendido.

Observando novamente o ator Participante de Reunião (figura 18), verificamos que existem várias dependências entre este ator e o ator Agendador de Reunião (tarefa EntrarDatasDisponíveis, recurso Concordância, etc) o que caracteriza-o como importante em um contexto de interação com o sistema pretendido. Portanto, o mesmo possui características que permitem que Participante de Reunião seja um ator no Diagrama de Casos de Uso.

Diretriz 4: em relacionamentos do tipo **ISA** entre atores genéricos tais como: o **ator1** é (ISA) do tipo **ator2,** ambos os atores devem ser avaliados conforme diretrizes anteriores. Normalmente, ambos os atores são considerados atores em um Diagrama de Casos de Uso. Deve-se observar também que este tipo de relacionamento é modelado no Diagrama de Casos de Uso através do relacionamento ***<<generalization>>*** (generalização)**,** enquadrando-se o ator 2 como uma especialização do ator 1. A notação no diagrama de casos de uso para este tipo de relacionamento entre atores foi apresentada na seção 3.2.

Por exemplo, na figura 18, o relacionamento ISA entre Participante Importante e Participante de Reunião indica que ambos os atores são candidatos a atores no diagrama de caso de uso. Conforme diretrizes anteriores já aplicadas na avaliação do ator Participante de Reunião, o mesmo pode ser considerado como ator em Caso de Uso. Da mesma forma, aplicando-se as mesmas diretrizes ao ator Participante Importante, pode-se apontar o mesmo como ator no Diagrama de Caso de Uso. O relacionamento entre ambos deve ser do tipo <<generalization>>. O ator Participante Importante deve ser anotado como uma especialização de Participante de Reunião.

As diretrizes acima permitem a elaboração de uma lista de atores para o modelo de casos de uso. Portanto, tendo os atores do sistema definidos devemos então encontrar os casos de uso do sistema associados com os mesmos. Assim, o próximo passo no processo de integração i* e Casos de Uso é encontrar possíveis candidatos a casos de uso dos atores definidos, seguido das descrições dos seus fluxos (cenário) principal e alternativos. Na próxima seção, descrevemos algumas diretrizes que podem auxiliar nesta tarefa.

5.2.2 Diretrizes para a descoberta de casos de uso

Para descobrirmos os casos de uso do sistema, devemos em um primeiro passo observar o Modelo de Dependências Estratégicas. Uma análise deste modelo permite-nos visualizar a possibilidade de a partir das dependências estratégicas associadas com atores em i*, derivar os Casos de Uso para estes atores. O modelo de Dependências Estratégicas representa, basicamente, dependências entre atores do tipo objetivo, recurso, tarefa e objetivo-soft. Como algumas destas dependências estão associadas a sistemas computacionais pretendidos pela organização, é razoável afirmar que estas dependências podem derivar casos de uso para estes sistemas. Além disso, encontramos também no modelo de dependências estratégicas, dependências do tipo objetivo-soft que na Engenharia de Requisitos são denominadas de requisitos não-funcionais, os quais por sua vez, podem ser associados ao sistema pretendido. Portanto, além dos Casos de Uso (funcionalidade) de sistemas, visualiza-se também a possibilidade de que aspectos não-funcionais possam ser associados com sistemas computacionais pretendidos.

A tarefa de analisar os modelos em i* e derivar informações para Casos de Uso não é trivial, pois ambas as técnicas permitem enfocar requisitos de software sob

diferentes perspectivas. O modelo de Casos de Uso é um modelo criado com o fim de representar as várias situações de uso do sistema e conseqüentemente, os requisitos funcionais do mesmo. Já a modelagem organizacional, permite que intenções e necessidades de atores da organização, em relação a sistemas computacionais, possam ser descritas. No entanto, como os modelos organizacionais podem incluir alternativas para a obtenção de objetivos organizacionais, uma das vantagens deste mapeamento é que o engenheiro de requisitos, além de estar envolvido conscientemente com os objetivos organizacionais pode também, juntamente com os demais *stakeholders*, identificar alternativas tais como diferentes sistemas computacionais que satisfaçam da melhor maneira possível as metas e objetivos da organização e/ou dos clientes. Assim, aspectos anteriormente não considerados na descoberta de Casos de Uso podem ser avaliados, analisados e incluídos na descrição dos mesmos.

É necessário tornar mais sistemático este processo de integração. Inicialmente, verificamos que os Objetivos (*Goals*) definidos como dependências entre um ator da organização e um sistema computacional pretendido (também representado como ator em i*), podem ser mapeados para casos de uso de sistemas computacionais. Isto ocorre porque estes objetivos representam metas que atores esperam alcançar e que por sua vez só podem ser alcançadas com o auxílio do sistema computacional (em virtude da dependência estabelecida). Alcançar uma meta ou objetivo envolverá interações entre um ator organizacional e o sistema pretendido. Assim, estes objetivos, podem originar casos de uso no sistema cuja principal contribuição para o usuário ou organização seja a realização do objetivo.

Os outros tipos de dependências apresentados em i* do tipo tarefa e recurso, existentes entre atores organizacionais e atores que representam sistemas computacionais, também podem derivar casos de uso de sistemas. Estas dependências

podem estar associadas a tarefas que usuários esperam que sistemas realizem, (e isto exige interação com usuários), ou à necessidade de obtenção de recursos (informações) com o auxílio de sistemas computacionais. A outra dependência em i* do tipo objetivo-soft representa, na verdade, requisitos não-funcionais que podem ser associados ao sistema pretendido. Salientamos que na próxima seção, descrevemos algumas diretrizes que permitem mapear objetivos-soft descritos nas decomposições internas dos atores nos Modelos de Razões Estratégicas (SR), para requisitos não-funcionais associados com casos de uso já descobertos para o sistema pretendido. No entanto, objetivos-soft descritos nos Modelos de Dependências Estratégicas (SD) são mapeados para requisitos não-funcionais do sistema de forma geral.

Desta forma, os elementos nos Modelos SD de i* (desde que envolvam o ator sistema computacional), são potencialmente candidatos a casos de uso do sistema computacional. É importante ressaltar que poderíamos estender nossa visão para avaliar dependências entre atores em i*, que não envolvam o sistema computacional. Isto nos levaria à descoberta e descrição de casos de uso de negócio. Este aspecto será tratado em trabalhos futuros.

A seguir, consideramos alguns questionamentos que podem nos auxiliar a mapear tarefas e recursos em i* para objetivos de sistemas computacionais, os quais podem ser candidatos em potencial a casos de uso:

1. se um ator depende ou deve satisfazer uma dependência do tipo **tarefa** em relação ao ator sistema computacional, deve-se investigar se essa tarefa necessita ser refinada/decomposta para ser realizada. A necessidade de decomposição caracteriza o mapeamento desta tarefa para um caso de uso, o qual conterá as sub-tarefas necessárias para realizar a tarefa.

2. se um ator depende ou deve satisfazer uma dependência do tipo **recurso** em relação ao ator sistema computacional, qual é o objetivo/finalidade da obtenção desse recurso no contexto de utilização do sistema computacional? A resposta para esta pergunta dará origem a um caso de uso que representará a necessidade de obtenção de um recurso por um ator via interação com o sistema computacional.

A seguir, descrevemos algumas diretrizes práticas que podem nos auxiliar na descobertas de casos de uso através de uma análise orientada a objetivos dos modelos organizacionais em i*.

Como apresentado na figura 17, a entrada para este passo é o Modelo de Dependências Estratégicas em i*. Para cada diretriz apresentada daremos um exemplo da aplicação da mesma, em relação ao problema agendamento de reuniões (figura 18) ou em relação ao problema de uma aplicação de comércio eletrônico (figura 20).

2º Passo: Descoberta de Casos de Uso

Como regra geral, adotamos que a descoberta de casos de uso para um ***ator(es)*** (descoberto(s) conforme diretrizes definidas na seção 5.2.1- 1º passo da proposta) ,envolve a análise de todas as dependências entre o ator em consideração e o ator que representa o sistema computacional pretendido. Inicialmente, observamos um ator descoberto como ***Dependee*** (sistema computacional → *dependum* → ator em consideração)**,** considerando quais dependências (*Dependum*) ele é responsável e deve atender para o *Depender* (ator sistema computacional).

Em um segundo momento, invertemos o foco da análise, observando as dependências atribuídas ao ator sistema computacional como *dependee* (ator em consideração → *dependum* → sistema computacional), em relação ao ator em caso de

uso analisado agora como *depender*. Cada *Dependum* (tarefa, recurso, etc) é candidato a Caso de Uso do ator. Adotamos também, que todos os *Dependum* que não são do tipo Objetivo, tais como: tarefa e recurso, serão analisados procurando relacioná-los a objetivos de sistemas computacionais. As dependências do tipo Objetivo-soft serão mapeadas para requisitos não funcionais do sistema pretendido. Dessa forma, temos:

Diretriz 5: para cada ator descoberto para o sistema (1^o passo da proposta), devemos observar todas as suas dependências (***dependum***) como ***dependee*** em relação ao ator que representa o sistema computacional pretendido (***depender***), visando descobrir casos de uso para o ator. Inicialmente, para efeitos de sistematização do mapeamento, bem como uma organização das informações relevantes, recomendamos criar uma tabela contendo os atores já descobertos para o modelo de casos de uso e adicionar a esta tabela as informações associadas com as dependências destes atores, observando os mesmos como *dependee* e o sistema computacional pretendido como *depender* (sistema computacional $\rightarrow$ *dependum* $\rightarrow$ ator em consideração). Neste ponto, também recomendamos adicionar à tabela, as informações sobre as dependências entre estes atores, agora observando os atores descobertos como ***depender*** e o ator sistema computacional pretendido como ***dependee*** (ator em consideração $\rightarrow$ *dependum* $\rightarrow$ sistema computacional). Esta informação será particularmente útil para aplicar a diretriz 6. É interessante também, adicionar qual heurística será aplicada para analisar a potencialidade de uma dependência originar um caso de uso. Esta tabela será bastante útil para poder coletar a informação que nos interessa dos Modelos de Dependências Estratégicas (SD) e aplicar as heurísticas apropriadas para analisar cada uma das dependências relevantes. A tabela 2 apresenta um exemplo de como as informações seriam apresentadas.

Ator	Dependência	Tipo de Dependência	Diretriz a ser usada
Banco	Processar Transações On-line	Objetivo	(D 5.1)

Tabela 2. Informação coletada a partir dos Modelos de Dependências Estratégicas visando auxiliar engenheiros de requisitos a derivar casos de uso.

Diretriz 5.1: dependências do tipo **objetivo -** cada dependência deste tipo deve ser mapeada diretamente para caso de uso do ator. Salientamos que estamos adotando que cada caso de uso possui um objetivo que representa o que deseja ser alcançado (Cockburn 2000). Assim, a dependência do tipo objetivo dará o nome ao caso de uso, bem como definirá qual objetivo espera-se alcançar com sua realização. Posteriormente, associaremos com cada objetivo um tipo (objetivo de contexto, objetivo de usuário, objetivo de subfunção), que identificará o nível de abstração do mesmo, auxiliando desta forma, na compreensão e entendimento do caso de uso associado.

No modelo da figura 18, infelizmente não temos nenhuma dependência do tipo objetivo entre o ator Agendador de Reunião (visto como depender) e outros atores (vistos como dependees). Contudo, no exemplo da figura 20 (capítulo 6), a dependência de objetivo Processar Transações On-Line entre o ator Medi@ (depender) e o ator Banco (dependee), pode ser mapeada para o caso de uso Processar Transações On-Line, o qual conterá os vários passos seguidos por Banco para lidar com as transações financeiras on-line.

Diretriz 5.2: dependências do tipo **tarefa -** verificar se para realizar a tarefa existe a necessidade de que a mesma seja decomposta em outras sub-tarefas. Isto implica em afirmar que o ator necessita realizar certa tarefa através da interação com o sistema computacional e este processo de interação envolve vários passos

que podem ser descritos em um caso de uso. Assim, se a tarefa claramente inclui outros passos para levá-la a cabo e necessita ser decomposta em passos do caso de uso devido a sua complexidade, a mesma deve ser mapeada para caso de uso.

Por exemplo, para a dependência do tipo tarefa EntrarIntervaloDatas(m) associada com ator Iniciador de Reunião (ver figura 18), podemos considerar que a tarefa do iniciador de fornecer um intervalo de datas dentro do qual uma reunião deve ser agendada, pode incluir vários aspectos (sub-tarefas que mais tarde serão mapeadas para passos do caso de uso derivado desta tarefa), tais como associar intervalos de datas com reuniões especificas, estabelecer prioridades para reuniões específicas, etc. Assim, a partir da tarefa EntrarIntervaloDatas(m), podemos gerar o caso de uso denominado EntrarIntervaloDatas para o ator Iniciador de Reunião.

Vale lembrar que, no Modelo de Razões Estratégicas são apresentadas as decomposições das ações que o ator deve levar a cabo para satisfazer cada *dependum* associado com o mesmo, do ponto de vista do ator como *dependee*. Assim, estão associados neste modelo, tarefas, objetivos, recursos e objetivo-soft que o ator internamente deve realizar para satisfazer as dependências com outros atores (*dependee*). No nosso caso, estas informações são usadas para detectar possíveis novos objetivos, (casos de uso), não expressos no Modelo de Dependências Estratégicas, bem como principalmente possíveis caminhos/passos em cenários de casos de uso já descobertos. Na diretriz 6, tratamos detalhadamente deste último aspecto.

Diretriz 5.3: dependências do tipo **recurso -** a pergunta chave para definir se uma dependência de recurso irá gerar ou não um caso de uso é investigar qual é o objetivo da obtenção deste recurso pelo ator ou ainda para que fim este recurso é

necessário? Se houver um objetivo maior, certamente este objetivo originará um caso de uso para o ator. O que ocorre na prática é que, normalmente, o nome do recurso é representativo do objetivo que se deseja alcançar com a obtenção/disponibilidade do mesmo, e assim este nome é mapeado diretamente para nome do caso de uso.

Analisando novamente a figura 18, do ponto de vista do ator Participante de Reunião como dependee, podemos observar a dependência do tipo recurso Concordância(m,p). Verificamos que o principal objetivo da obtenção deste recurso é obter dos participantes da reunião uma concordância em relação a uma data proposta para uma reunião. Diante deste objetivo, poderíamos considerar que no processo de obtenção de concordância, por exemplo, cada participante poderia concordar com a data proposta com certas restrições de tempo de duração da reunião ou mesmo do horário agendado. Ainda, a concordância poderia envolver uma análise de outras datas alternativas. Em outras palavras, para se obter uma concordância em relação a uma data proposta de agendamento de reunião, várias interações (passos) são necessárias entre o sistema agendador de reuniões e os participantes de reunião, sendo que estas interações poderiam ser especificadas em um caso de uso denominado Concordância atribuído ao ator Participante de Reunião.

Diretriz 5.4: dependências do tipo **objetivo-soft -** normalmente este tipo de dependência é associado na modelagem organizacional a uma meta que ainda não está totalmente bem definida (por exemplo, que uma reunião na organização seja rápida e produtiva). Portanto, um objetivo-soft corresponde a um requisito não-funcional associado ao sistema computacional e que pode eventualmente ser associado a um caso de uso específico (isto é, ser operacionalizado). Contudo,

como casos de uso são mais adequados para descrever requisitos funcionais, temos adotado neste trabalho descrever as dependências de objetivo-soft no Modelo de Dependências Estratégicas como requisitos não-funcionais associados com o sistema de forma geral.

Contudo, se temos o caso no qual um objetivo-soft é um sub-componente em uma decomposição de uma tarefa no Modelo de Razões Estratégicas, haverá uma exceção. Neste caso, se a tarefa decomposta no Modelo de Razões Estratégicas satisfaz alguma dependência mapeada para caso de uso, este objetivo-soft será associado com este caso de uso. Este aspecto é melhor descrito, inclusive com exemplificação, na diretriz 8, a qual lida mais especificamente com a análise dos Modelos de Razões Estratégicas. Salientamos também, que futuramente pretendemos analisar de que forma podemos utilizar na nossa proposta, o *framework* NFR apresentado em Chung et al. (2000), já que consideramos o nosso tratamento de requisitos não-funcionais ainda insatisfatório.

Por exemplo, na figura 20 (capítulo 6), podemos observar duas dependências do tipo objetivo-soft: Segurança e Disponibilidade entre Cliente e o sistema Medi@. Assim, neste caso, associamos ao Sistema Medi@ a necessidade expressa pelo cliente de que os serviços de transação sejam seguros e disponíveis.

Diretriz 6: as diretrizes anteriores conduzem-nos na análise de todos os atores descobertos para o sistema, do ponto de vista dos mesmos como *dependee*. No entanto, o ator em i*, que representa um sistema computacional, ainda não foi considerado nesta análise. A análise das dependências do ator sistema computacional em i* (agora como *dependee*), pode-nos levar a importantes casos de uso do sistema. Assim, devemos analisar esta situação especial, na qual um ator descoberto para o modelo de casos de uso (observado como *depender*), possui dependências em relação ao ator em i*, que

representa o sistema computacional pretendido (observado como *dependee*). Desta forma, observamos agora, relacionamentos do tipo ator → *dependum* → sistema computacional. Estas dependências devem ser analisadas em um mapeamento para casos de uso, utilizando as mesmas heurísticas 5.1, 5.2, 5.3 e 5.4, aplicadas quando o ator em casos de uso era visto como *dependee*. No entanto, podemos notar que nesta situação, o caso de uso derivado é associado com o ator observado como *depender* no relacionamento. Isto ocorre porque o *dependee,* nesta situação, é o ator sistema de software e o *depender*, (ator no modelo de caso de uso), deve interagir com o sistema para obter um objetivo associado com o caso de uso gerado. Além disso, outro aspecto importante que auxilia no mapeamento, é que o sistema pretendido, através de razões estratégicas internas, procura satisfazer as dependências atribuídas ao mesmo. Esta informação não está disponível no Modelo de Dependências Estratégicas (SD), mas pode ser observada no Modelo de Razões Estratégicas (SR). No modelo de Razões Estratégicas , as dependências sob responsabilidade do ator sistema computacional, devem ser satisfeitas através de ligações de decomposição de tarefa e/ou ligações meio-fim. A partir da observação destas ligações, podemos reforçar e confirmar a idéia de criação de um caso de uso originado de uma dependência que o sistema deve satisfazer. O caso de uso gerado conterá as sub-tarefas e sub-objetivos (razões internas representadas no Modelo de Razões Estratégicas) que o sistema computacional, (interagindo com o ator do caso de uso), necessita realizar para satisfazer uma dependência específica.

Para exemplificar como podemos aplicar esta diretriz, vejamos na figura 18, a dependência do tipo objetivo ReuniãoAgendada(m) entre o ator Iniciador de Reunião (depender) e o ator Agendador de Reunião (dependee). Este relacionamento atribui ao sistema agendador de reunião, o objetivo de realizar o

agendamento de reuniões. O objetivo ReuniãoAgendada é mapeado para caso de uso conforme diretriz 5.1. Este mapeamento é reforçado se observarmos, por exemplo, que existe uma dependência entre estes atores que indica que o iniciador de reunião deve fornecer inicialmente um intervalo, dentro do qual uma reunião deve ser agendada. Além disso, observando o modelo de Razões Estratégicas da figura 19, verificamos que depois de fornecido o intervalo para agendamento, o agendador de reunião realizará o agendamento através de uma série de passos (subtarefas e/ou subobjetivos). Estes passos incluem a solicitação de datas possíveis de agendamento dos participantes da reunião, com base no intervalo fornecido pelo iniciador, definição das melhores datas possíveis de agendamento, a partir do cruzamento das datas fornecidas pelos participantes, proposta de uma data específica para o agendamento e finalmente solicitação da concordância explícita dos participantes em relação à data de agendamento proposta. Assim, estas características reforçam a idéia de definição de um caso de uso denominado de ReuniãoAgendada, (seguindo o mesmo nome atribuído a dependência do tipo objetivo que origina o caso de uso), para o ator Iniciador de Reunião. Este caso de uso especifica os principais passos do processo de agendamento de reuniões, (a diretriz 8 lida com a questão de como derivar passos de casos de uso a partir dos Modelos SR).

Diretriz 7: após a descoberta dos casos de uso iniciais, é aconselhável classificar cada caso de uso com o tipo associado a seu objetivo (contextual, objetivo de usuário ou subfunção). Salientamos que adotaremos um *template* para especificação de caso de uso derivado de (Cockburn 2000), no qual um dos campos refere-se ao ***Nível*** do objetivo do caso de uso (ver figura 5). O fato de associar um objetivo a um caso de uso, bem como

classificar este objetivo em um dos níveis propostos (Cockburn 2000), auxilia-nos em vários aspectos, conforme segue:

1. definir um objetivo para o caso de uso nos proporciona saber exatamente o que se quer obter com a sua realização;
2. definir o objetivo do caso de uso auxilia-nos na decisão de quando parar de descrevê-lo;
3. conhecer o objetivo permite-nos também investigar alternativas associadas com possíveis condições de exceções ou falhas, oriundas da análise dos impedimentos (sob quais condições o objetivo pode falhar?) na obtenção do objetivo de caso de uso;
4. ter a estrutura de objetivos com casos de uso é bastante útil para o gerenciamento e rastreamento de projetos;
5. existem, principalmente, dois aspectos que causam confusão para as pessoas que lidam com casos de uso. O primeiro aspecto envolve detectar e definir adequadamente o escopo ("*boundary*") do caso de uso. O segundo aspecto (o qual tratamos nesta diretriz), envolve definir a especificidade ou nível de abstração do objetivo do caso de uso. Constantemente, desenvolvedores questionam-se a respeito de qual é o nível apropriado do objetivo que o caso de uso deveria descrever. A proposta apresentada nesta diretriz de associar um nome de nível (definição de abstração do que se deseja alcançar) com o objetivo de caso de uso, minimiza esta confusão. Conhece-se melhor, assim, o que se está descrevendo no caso de uso. A classificação que adotamos para objetivo de caso de uso tem como base a proposta de Cockburn (2000), a qual classifica os objetivos em objetivo de contexto, usuário e subfunção.

Para auxiliar engenheiros de requisitos a identificar novos casos de uso, bem como compreender melhor os casos de uso definidos, recomendamos criar uma tabela contendo o nome do ator, o objetivo do caso de uso e a classificação do nível do objetivo. (veja tabela 3).

Ator	Objetivo do Caso de Uso	Classificação do Objetivo
Media Shop	Processar Pedidos da Internet	Objetivo de Contexto

Tabela 3. Classificação de objetivos de casos de uso.

É importante também salientar que cada passo na descrição (cenários primário e secundário) dos casos de uso originados destes objetivos, poderá representar objetivos do mesmo nível ou de um nível abaixo (seguindo a ordem objetivo de contexto, objetivo de usuário e objetivo de subfunção), os quais poderão originar outros casos de uso para o sistema. Estes relacionamentos poderão ser expressos via mecanismos do tipo <*include*>, <*extend*> e <*generalization*>. Este aspecto será melhor descrito e tratado na diretriz 8.

Assim, com base na classificação de objetivos proposta, deve-se enquadrar os objetivos dos casos de uso previamente descobertos.

Nesta seção, apresentamos as diretrizes propostas para derivar casos de uso a partir dos modelos organizacionais em i*. Um aspecto importante a ressaltar é que este processo de derivação é realizado através de uma análise orientada a objetivos, a qual está fundamentada nos conceitos apresentados na Engenharia de Requisitos Orientada a Objetivos (*GORE – Goal Oriented Requirements Engineering*). As técnicas (Anton 1997), (Potts 1999) (Kaiva et al. 2002), que pertencem a este enfoque da Engenharia de Requisitos, visam basicamente a descoberta de requisitos de sistemas através da elicitação, especificação e operacionalização de objetivos de usuários em relação a

sistemas computacionais. No nosso caso, o estudo destas técnicas fundamentou a criação das heurísticas para derivar objetivos de casos de uso a partir dos elementos representados nos Modelos de Dependências Estratégicas (SD).

Outro aspecto importante na nossa proposta (como comentado anteriormente), recai no fato de que após a descoberta dos casos de uso e descrição dos mesmos no passo 3 da proposta (utilizando os Modelos de Razões Estratégicas (SR)), poderemos ainda derivar novos casos de uso a partir das descrições de cada passo dos fluxos principal e alternativos das especificações dos casos de uso já descobertos. Isto exigirá uma análise complementar com a utilização, inclusive, de algumas heurísticas adicionais derivadas do método *GBRAM* (Anton 1997), buscando definir se um determinado passo na especificação de um caso de uso pode originar um novo caso de uso. Temos optado por utilizar algumas heurísticas do método *GBRAM,* por considerar que as mesmas propiciam o enriquecimento da nossa proposta, possibilitando a descoberta de casos de uso importantes de sistemas computacionais (ver seção 3.3).

A seguir, definimos as diretrizes que visam apoiar o processo de especificação/descrição dos casos de uso descobertos utilizando as diretrizes apresentadas nesta seção.

5.2.3 Diretrizes para a descrição/especificação dos casos de uso

Continuando o processo de integração de casos de uso e i*, conforme descrito na figura 17, o passo 3 lida com a especificação dos casos de uso descobertos observando-se os Modelos de Razões Estratégicas (SR). Lembremos que os modelos SR em i* descrevem as razões estratégicas internas de atores relacionadas com processos organizacionais. Estas razões estratégicas são representadas através de ligações do tipo meio-fim e decomposição de tarefas (ver seção 4.1.2), as quais visam descrever de

forma mais detalhada passos para obtenção de objetivos, recursos, tarefas e objetivos-soft de um ator. Com base nestas ligações, podemos extrair os passos dos cenários primários e secundários (fluxos principal e alternativos) de casos de uso. A seguir, descrevemos a diretriz que orienta este processo de especificação de casos de uso.

3º Passo: Especificação de Casos de Uso

Diretriz 8: deve-se analisar cada ator e seus relacionamentos no Modelo de Razões Estratégicas, visando derivar informação que possa conduzir à descrição dos cenários primário e secundários dos casos de uso descobertos. Inicialmente, verificamos que cada caso de uso e seu objetivo associado são atribuídos a um ator do sistema, seguindo as diretrizes do passo 2. A tabela resultante da diretriz 7 (ver tabela 3), apresenta informações sobre o ator do sistema, o objetivo do caso de uso descoberto e o tipo do objetivo. O objetivo do caso de uso descoberto é oriundo de uma dependência no Modelo de Dependências Estratégicas, a qual é satisfeita/alcançada pelo ator *dependee* no relacionamento. Assim, observando o Modelo de Razões Estratégicas, devemos considerar os elementos internos que são utilizados pelo ator *dependee* para satisfazer objetivos e objetivos-soft, realizar tarefas e obter recursos. O ator tem a responsabilidade de satisfazer estes elementos (dependências) e a decomposição no Modelo de Razões Estratégicas mostra-nos como o ator fará isto. Tipicamente, as dependências associadas com o ator (*dependee*) são satisfeitas internamente através de dois tipos de relacionamentos usados em SR: **ligações de decomposição de tarefa e ligações meio-fim**. Estes relacionamentos são observados para derivar passos na descrição de um caso de uso. Essas ligações são mapeadas conforme segue:

> **Diretriz 8.1:** sub-componentes em uma ligação de ***decomposição de tarefa*** são mapeados para passos na descrição do cenário primário (fluxo principal)

de um caso de uso originado da dependência que a tarefa decomposta satisfaz. Neste caso, a tarefa decomposta no Modelo de Razões Estratégicas é responsável por satisfazer uma dependência já mapeada para caso de uso. A única exceção está no sub-componente que representa um objetivo-soft. Este sub-componente será mapeado para um requisito não-funcional do caso de uso que a tarefa observada e decomposta realiza/satisfaz. No *template* de escrita de caso de uso da figura 5, este objetivo-soft pode ser descrito no campo denominado Requisitos Especiais.

Outro aspecto importante neste mapeamento, está na observação de que os sub-componentes da tarefa decomposta podem ter ligações de dependências (já estabelecidas no Modelo de Dependências Estratégicas, mas representadas de forma mais específica em relação a um componente no Modelo de Razões Estratégicas) com outros sub-componentes de outros atores. Estas dependências, como já foram definidas no Modelo de Dependências Estratégicas, podem ter sido mapeadas também para casos de uso seguindo as diretrizes do passo 2 da nossa proposta. Sendo assim, o que ocorre neste caso é que o sub-componente mapeado para um passo no caso de uso, também estabelece a necessidade de incluir, via mecanismo <<*include*>> descrito na seção 3.1, o caso de uso que foi originado da dependência associada. Isto na prática quer dizer que, determinado sub-componente mapeado para um passo em caso de uso está relacionado a uma tarefa ou objetivo que é complexo e importante o suficiente para ser descrito e refinado em um novo caso de uso. Como na verdade a dependência já tem sido mapeada para caso de uso, basta apenas utilizar o mecanismo <<*include*>> no passo do caso de uso.

Por exemplo, na figura 18 existe uma dependência do tipo objetivo, denominada ReuniãoAgendada, entre o ator Iniciador de Reunião e ator Agendador de Reunião. Esta dependência havia sido previamente mapeada para caso de uso seguindo a diretriz 6. O que estamos procurando neste momento são as razões internas do ator Agendador de Reunião (dependee no relacionamento) para satisfazer esta dependência. Estas razões são observadas no Modelo de Razões Estratégicas da figura 19, no qual a tarefa interna ao sistema agendador de reunião denominada de AgendarReuniãoAutomaticamente, é responsável por obter o objetivo ReuniãoAgendada para o ator Iniciador de Reunião. Esta tarefa, no entanto, é decomposta nos sub-componentes: ObterDatasDisponíveis, EncontrarMelhorData, ProporData e ObterConcordância, os quais são mapeados para os passos de alto nível da especificação do caso de uso ReuniãoAgendada, mostrando como o sistema agendador de reunião procederá o agendamento de reuniões, interagindo com o iniciador de reunião e outros atores do sistema. Em uma segunda análise, verificamos também que, o sub-componente ObterDatasDisponíveis tem uma dependência direta associada (já definida para o ator no Modelo de Dependências Estratégicas da figura 18) do tipo tarefa denominada de EntrarDatasDisponíveis, em relação ao ator Participante de Reunião. Vejamos que esta dependência, conforme diretriz 5.2 origina um caso de uso de nome EntrarDatasDisponíveis. O que ocorre nesta situação é que no passo do caso de uso gerado pelo sub-componente ObterDatasDisponíveis haverá um relacionamento do tipo <<include>> para o caso de uso

EntrarDatasDisponíveis, expressando a idéia de que esta tarefa será refinada e descrita em outro caso de uso.

Outro exemplo que pode mostrar o mapeamento de sub-componentes de uma tarefa para passos de caso de uso, bem como mostrar o mapeamento de um sub-componente do tipo objetivo-soft para requisito não-funcional do caso de uso, pode ser observado na figura 21. A tarefa interna ao ator Medi@ denominada Gerenciar Loja da Internet, satisfaz a dependência Processar Pedidos da Internet para o ator Media Shop. Esta tarefa é decomposta nos seguintes sub-componentes: os sub-objetivos Tratar Pesquisa de Item e Tratar Pedidos da Internet; a sub-tarefa Produzir Estatísticas e o objetivo-soft Atrair Novos Clientes. Assim, estes sub-componentes, com exceção do objetivo-soft Atrair Novos Clientes, são mapeados para o cenário primário (fluxo principal) do caso de uso Processar Pedidos da Internet. Já o objetivo-soft Atrair Novos Clientes é definido (no campo requisitos especiais do template de caso de uso mostrado na figura 5) como um requisito não-funcional associado com o caso de uso Processar Pedidos da Internet.

Diretriz 8.2: ligações do tipo meio-fim representam, basicamente, um meio (normalmente uma tarefa) para atingir um fim (que pode ser um objetivo, objetivo-soft, recurso ou mesmo tarefa). O meio, tipicamente, representa uma alternativa para se atingir um fim. Se este fim é um sub-componente em uma ligação de decomposição de tarefa, devemos observar se tarefa decomposta satisfaz (conforme diretriz 8.1) uma dependência mapeada para caso de uso. Se isto ocorre, o meio analisado, na verdade é um caminho para

realizar o sub-componente mapeado para um passo do caso de uso. Se existir apenas um meio para se atingir o fim (sub-componente), então este meio deve complementar a descrição do passo do caso de uso que o sub-componente origina. Por outro lado, se existirem mais de um meio para se atingir um fim (sub-componente que é um passo do caso de uso), deve-se optar por um meio que representa o caminho normal (descrito no cenário primário – fluxo principal do caso de uso) a ser seguido, e os outros meios mapeados para passos alternativos ou "*extends*" do passo que o sub-componente origina.

Para entender melhor este mapeamento, voltemos ao exemplo apresentado na diretriz 8.1. Vimos que a tarefa na figura 19 AgendarReuniãoAutomaticamente é decomposta nos sub-componentes: ObterDatasDisponíveis, EncontrarMelhorData, ProporData e ObterConcordância, os quais são mapeados para os passos de alto nível da especificação do caso de uso ReuniãoAgendada. Se observarmos o sub-componente do tipo objetivo EncontrarDatasPossíveis (ver figura 19), o mesmo pode ser obtido pelo "meio" CruzarDatasDisponíveis. Como só existe este meio para atingir o fim (EncontrarDatasPossíveis), o passo no cenário primário do caso de uso seria acrescido deste meio estabelecendo que este é o caminho normal de realização adotado no caso de uso. Assim, a descrição do passo do caso de uso poderia ser "O sistema deve encontrar datas possíveis de agendamento através do cruzamento de datas disponíveis (CruzarDatasDisponíveis)". Se existissem outros meios para se alcançar o sub-objetivo

EncontrarDatasPossíveis, este meios seriam mapeados para "extends" ou passos alternativos (fluxos alternativos) do caso de uso. Um exemplo desta situação é apresentada no capítulo 6.

Diretriz 8.3: outro aspecto importante na especificação de casos de uso a partir dos Modelos de Razões Estratégicas (SR), é a possibilidade do mapeamento de *pré-condições* e *pós-condições de sucesso e falha* do caso de uso. Uma ***tarefa*** interna no Modelo de Razões Estratégicas (interna a um ator observado como *dependee*) que satisfaz uma dependência mapeada previamente para o caso de uso, pode conter também ligações de ***dependências*** em relação a outros atores. Estas dependências serão mapeadas para *pré-condições* do caso de uso (mapeado a partir de uma dependência observada no Modelo de Dependências Estratégicas), que a tarefa está satisfazendo. Esta mesma tarefa escrita no tempo verbal passado pode ser mapeada para uma *pós-condição de sucesso* do caso de uso. Além dessa tarefa, a outra pós-condição de sucesso óbvia do caso de uso é o próprio objetivo do caso de uso escrito no tempo passado. Já no caso de falha na obtenção do caso de uso, essa mesma tarefa expressa na forma negativa pode ser mapeada para *pós-condição de falha*. Novamente, a outra pós-condição de falha óbvia do caso de uso é a negativa do próprio objetivo do caso de uso (mapeado de uma dependência do modelo em i*).

Por exemplo, voltemos ao Modelo de Razões Estratégicas da figura 19. A tarefa AgendarReunião Automaticamente interna ao ator Agendador de Reunião tem a responsabilidade de satisfazer a dependência Reunião Agendada mapeada para caso de uso (usando diretriz 6). Esta tarefa é

decomposta em vários sub-componentes que representam passos no caso de uso derivado, mas também possui uma dependência direta denominada de EntrarIntervaloDatas em relação ao ator Iniciador de Reunião. Esta dependência do tipo tarefa será mapeada para uma pré-condição do caso de uso Reunião Agendada, estabelecendo que o iniciador de reunião precisaria já ter fornecido o intervalo de datas, dentro do qual uma reunião seria agendada. Por outro lado, a mesma tarefa AgendarReunião Automaticamente escrita no passado, pode ser mapeada para pós-condição de sucesso do caso de uso. Uma pós-condição de falha, para este caso de uso, pode ser definida negando-se a tarefa AgendarReunião Automaticamente. Uma aplicação mais completa e mais detalhada desta diretriz pode ser vista no estudo de caso de comércio eletrônico apresentado no capítulo 6.

Diretriz 9: nesta diretriz, lidamos com a possibilidade de derivar novos objetivos e 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. Isto pode ser feito através das seguintes investigações:

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; neste sentido a heurística HIG 1 (ver seção 3.3.1) proposta por Anton (1997), pode ser utilizada. Mais especificamente, a seguinte pergunta é bastante útil:

Qual objetivo(s) esta sentença/frase exemplifica?

*Observemos o caso de uso 2 (**Processar Pedidos da Internet),** gerado no estudo de caso de um sistema de comércio eletrônico (ver figura 25 no capítulo 6). O cenário primário (fluxo principal) do caso de uso **Processar Pedidos da Internet** possui três passos (previamente mapeados a partir de modelos organizacionais). O passo (sentença) número 3 é assim descrito: "o sistema produz estatísticas para Media Shop". Assim, caberia neste momento responder à pergunta "qual objetivo esta sentença exemplifica?". Parece coerente responder que o objetivo deste passo é **Produzir Estatísticas**.*

Diretriz 9.2: descoberto o objetivo, deve-se averiguar a necessidade de decompor/refinar o objetivo para que o mesmo possa ser alcançado; a seguinte pergunta é bastante útil:

Quais são os passos necessários (sub-objetivos) para se obter o objetivo?

*Continuando na análise do passo 3 no exemplo do caso de uso **Processar Pedidos da Internet** (ver figura 25 no capitulo 6), devemos investigar a resposta para a pergunta acima. Tipicamente, produzir estatísticas para Media Shop poderia envolver entre outros aspectos: definir o tipo de estatística de pedidos desejada podendo ser, por exemplo, pedidos de um item específico, pedidos de uma categoria específica de itens, pedidos no geral, etc; definir o período (dias, semanas,meses,ano,etc) para o qual as estatísticas serão produzidas; estabelecer sobre qual região geográfica especifica, por exemplo, quer se produzir dados estatísticos. Enfim, pode-se definir uma série de sub-objetivos (passos) que seriam necessários para poder alcançar o objetivo do passo 3 "Produzir Estatísticas"do caso de uso observado.*

Diretriz 9.3: um novo caso de uso será gerado a partir do objetivo analisado (resposta à pergunta da diretriz 9.1), se pudermos definir os passos que representam a necessidade de interações adicionais entre o ator e o sistema para se atingir o objetivo desejado.

Assim, como temos definido no exemplo da diretriz 9.2 a necessidade de vários passos para se atingir o objetivo do passo 3 do caso de uso observado, podemos apontar a necessidade de gerar um caso de uso denominado ***Produzir Estatísticas****, o qual incluirá os passos de interação entre Media Shop e o sistema Medi@ para produzir estatísticas de pedidos.*

Diretriz 9.4: adicional e opcionalmente, as heurísticas proposta por Anton (1997) no método GBRAM, e descritas na seção 3.3.1, podem ajudar a encontrar novos objetivos e cenários com base na observação dos cenários e obstáculos de objetivos já descobertos. No entanto, isto implica em praticamente aplicar a proposta de Anton (1997) e, portanto, em um esforço extra importante, mas que será melhor investigado em trabalhos futuros.

Diretriz 10: concluídas as investigações para derivar atores e casos de uso, bem como especificar os casos de uso descobertos, podemos gerar uma versão (gráfica) do diagrama de caso de uso conforme a notação adotada pelo UML (atualmente na versão 1.4.1), para o sistema a ser desenvolvido. Na seção 3.1 foram apresentados os elementos essenciais para construção deste diagrama.

Neste trabalho, utilizamos os relacionamentos estratégicos entre atores definidos em i* como fonte de origem de casos de uso de sistemas computacionais. Motiva-nos, entre outros aspectos, que esta estratégia poderia também ajudar a solucionar alguns dos principais problemas existentes no processo de construção de casos de uso. Alguns destes problemas são expressos em (Lilly 1999). Apresentamos abaixo os problemas

juntamente com alguns sintomas associados com os mesmos. Apresentamos também uma possível solução para cada problema utilizando a nossa proposta:

Problema 1: o limite do sistema está indefinido ou inconsistente;

Sintoma: os casos de uso são descritos em um escopo de sistema inconsistente – alguns casos de uso são definidos a nível de escopo de negócio, outros; a nível de sistema ou subsistema em um mesmo modelo/diagrama de casos de uso.

Possível Solução: na nossa proposta derivamos casos de uso de sistemas computacionais, observando diretamente as dependências entre atores da organização e o ator em i*, que representa o sistema computacional pretendido. Isto nos garante que os relacionamentos estratégicos mapeados para casos de uso sejam provenientes de um desejo explicitamente modelado em i*, entre um ator organizacional e o sistema pretendido. Todos os casos de uso derivados nesse processo estão dentro do limite do sistema computacional pretendido. Por outro lado, casos de uso derivados de relacionamentos que envolvem atores da organização e que não incluem o ator sistema computacional, podem ser mapeados para casos de uso de negócio (caso haja interesse e cabe aí deixar isto explícito e definir em um modelo/diagrama de casos de uso próprio), sendo definidos assim, consistentemente, fora do limite do sistema computacional. Isto permite que possamos definir e tornar consistentes os limites do sistema desejado. Além disso, temos optado por utilizar o *template* de especificação de caso de uso (ver figura 5) proposto por Cockburn (2000). Neste *template* existe um campo específico para anotar o escopo do caso de uso.

Problema 2: os casos de uso são escritos do ponto de vista do sistema e não dos atores do sistema;

Sintoma: os nomes dos casos de uso descrevem o que o sistema faz, ao invés de descrever o objetivo que o ator possui em relação ao caso de uso.

Possível Solução: casos de uso na nossa proposta são derivados diretamente da dependência de atores do sistema em relação ao ator sistema pretendido. Isto garante que um caso de uso está associado e será relevante para satisfazer um objetivo ou meta de um ator do sistema. Este objetivo será modelado e descrito explicitamente na especificação do caso de uso.

Problema 3: os nomes de atores são inconsistentes;

Sintoma: diferentes nomes de atores são usados para descrever o mesmo papel.

Possível Solução: quando aplicamos as heurísticas definidas na proposta, na qual atores do sistema são derivados a para a partir de atores em i*, temos a oportunidade de observar e eliminar nomes de atores inconsistentes.

Problema 4: existem muitos casos de uso;

Sintoma: O modelo de caso de uso possui um número muito grande de casos de uso;

Possível Solução: este problema pode ser solucionado em uma primeira instância, se observamos que os casos de uso derivados na nossa proposta são provenientes de relacionamentos entre atores organizacionais, os quais representam as dependências "estratégicas", ou seja, mais importantes ou essenciais do ponto de vista destes atores. Estas dependências, consideradas estratégicas, derivam os casos de uso mais importantes do sistema. Em um primeiro passo, usando as heurísticas propostas, o risco de se definir muitos casos de uso é bastante minimizado.

Problema 5: os relacionamentos de atores para casos de uso assemelham-se a uma teia de aranha;

Sintoma: (a) existem muitos relacionamentos entre atores e casos de uso. (b) um ator interage com todos os casos de uso. (c) um caso de uso interage com todos os atores.

Possível solução: o fato de associar um ator do sistema com casos de uso derivados de dependências estratégicas, minimiza os riscos de se associar um ator com um número não consistente (dependências inexistentes em i*) e excessivo de casos de uso. Tipicamente, um ator possui um conjunto limitado de dependências estratégicas em relação ao sistema computacional.

Problema 6: as especificações/descrições de casos de uso são muito extensas;

Sintoma: a especificação de um caso de uso envolve mais de uma página.

Possível Solução: as descrições básicas de casos de uso na nossa proposta tomam como base as ligações de decomposição de tarefa e ligações meio-fim no modelo de Razões Estratégicas, as quais são mapeadas para o fluxo principal de casos de uso e fluxos alternativos, respectivamente. As ligações utilizadas em i* mostram normalmente as atividades (tarefas ou objetivos) consideradas essenciais na realização de uma tarefa, o que minimiza a possibilidade de descrições muito extensas. Reduz-se, significantemente, também o risco de incluir detalhes excessivos em conseqüência de uma visão voltada para aspectos de interface ou implementação, pois o modelo de Razões Estratégicas em i* visa representar as razões estratégicas de atores que por definição (Yu 95), não devem envolver aspectos de implementação ou interface.

Problema 7: as especificações/descrições de casos de uso são muito confusas;

Sintoma: falta contexto ao caso de uso; este não "conta uma história".

Possível Solução: este problema é comum quando não contextualizamos o caso de uso sendo descrito e/ou descrevemos aspectos de implementação ao invés de

descrever interações externas. Inicialmente, temos optado por utilizar o *template* de especificação de caso de uso proposto por Cockburn (2000), o qual contém o campo *Objetivo no Contexto* descrevendo em que contexto o objetivo do caso de uso deseja ser alcançado, facilitando assim a sua compreensão. A derivação dos passos de casos de uso, a partir de ligações de decomposição de tarefa, permite organizar o caso de uso com os passos essenciais. Como mapeamos ligações meio-fim para caminhos alternativos de casos de uso, o uso de "se (*if*) então" que tornam o caso de uso confuso também é bastante reduzido.

Problema 8: os casos de uso não descrevem corretamente o objetivo funcional associado;

Sintoma: as associações entre atores e casos de uso não descrevem correta ou completamente "quem" pode fazer "o quê" no sistema.

Possível Solução: este problema ocorre quando o ator não pode realizar ou ter acesso a toda a funcionalidade descrita no caso de uso, tendo o acesso restrito apenas a uma parte do caso de uso. Comumente, casos de uso muito gerais, podem ter este tipo de problema. A nossa proposta parte de dependências de atores para derivar casos de uso, o que representa a necessidade de alcançar determinado objetivo/meta por parte do ator. No entanto, o problema de acesso neste passo descrito ainda pode ocorrer e necessita de uma análise complementar, visando garantir que toda a funcionalidade descrita no caso de uso deve ser realizada pelo ator. Caso contrário, haverá a necessidade de dividir o caso de uso com esse problema, visando separar a funcionalidade que pode ser realizada completamente por um ator (veja exemplo em (Lilly 1999)).

Problema 9: o cliente não compreende os casos de uso;

Sintoma: o cliente não compreende os casos de uso e necessita revisar ou aprovar um documento de requisitos baseado em casos de uso.

Possível Solução: como casos de uso são derivados de modelos organizacionais previamente acordados entre os atores da organização, derivar casos de uso a partir destes modelos facilita o entendimento por parte dos atores (clientes), pois houve uma integração anterior ao processo de criação de casos de uso, na qual clientes expressaram e modelaram em i* as suas intenções e razões estratégicas em relação a sistemas pretendidos. i* permite conhecer o negócio, bem como as intenções sobre sistemas computacionais. Além disso, a definição do **contexto** de realização do caso de uso que auxilia na compreensão dos mesmos é explicitamente especificado na nossa proposta em um campo previamente definido. Outro aspecto positivo, é que casos de uso são descritos em linguagem próxima do dia-a-dia de clientes (ao contrário de casos de uso descritos em linguagem adaptada ao computador e incompreensível, muitas vezes, por usuários), pois os elementos que originaram as descrições foram expressos pelos próprios clientes e engenheiros de requisitos nos modelos em i*. Finalmente, o fato de derivarmos os passos de casos de uso a partir de ligações de decomposição de tarefas e ligações meio-fim, as quais são informações estratégicas (essenciais) na organização, torna casos de uso mais concisos e mais fáceis de serem entendidos por clientes.

Problema 10: os casos de uso nunca são concluídos.

Sintoma: casos de uso precisam ser modificados toda vez que as interfaces de sistemas mudam.

Possível Solução: casos de uso derivados de dependências estratégicas são menos propensos a incluírem aspectos de interfaces e assim *depender*em menos

de aspectos de projeto, o que provoca que os mesmos nunca sejam concluídos. Casos de uso devem descrever interações externas entre o sistema e o cliente e não envolver aspectos de projeto ou implementação. Projetos de Interface devem satisfazer os requisitos definidos em casos de uso. Na nossa proposta, casos de uso são derivados de objetivos estratégicos e são independentes de aspectos de projeto ou implementação.

Desta forma, os itens acima resumem alguns problemas com casos de uso que poderiam ser minimizados adotando o nosso mapeamento.

5.3 Considerações Finais

Neste capítulo, apresentamos as diretrizes que auxiliam no processo de derivação de casos de uso a partir dos modelos organizacionais em i*. O que é importante ressaltar é que nesta abordagem o engenheiro de requisitos pode iniciar o desenvolvimento de casos de uso já familiarizado com intenções e objetivos organizacionais estratégicos. O fato de se desenvolver um modelo organizacional no processo de Engenharia de Requisitos possibilita esta visão bastante útil. É importante também observar que alguns componentes definidos nos modelos em i*, não estão associados diretamente a funcionalidades de sistemas computacionais pretendidos e, portanto, não são incluídos no nosso processo de mapeamento de i* para casos de uso. No entanto, entendemos que estas informações são essenciais para o entendimento do ambiente organizacional, no qual o software irá operar bem, como para avaliar as mudanças que ocorrerão nos processos organizacionais em virtude do novo software a ser desenvolvido. No próximo capítulo, apresentamos dois estudos de caso, visando uma melhor compreensão da nossa proposta: um mostrando o exemplo do problema de

agendamento de reuniões e outro mostrando um exemplo de aplicação de comércio eletrônico.

Capítulo 6

Estudos de Caso

Neste capítulo, aplicamos a nossa proposta de derivação de casos de uso a partir de modelos organizacionais em i* a dois estudos de caso: o primeiro é uma aplicação de comércio eletrônico e o segundo é o problema bastante conhecido de agendamento de reuniões. Inicialmente, na seção 6.1, a nossa proposta é aplicada ao estudo de caso de comércio eletrônico para uma empresa denominada Media Shop. Os modelos de Dependências Estratégicas e de Razões Estratégicas, utilizados para a descoberta e especificação dos casos de uso para este sistema, são extraídos de Castro et al. (2002). Na seção 6.2, aplicamos a nossa proposta ao problema de agendamento de reuniões. Os modelos de Dependências Estratégicas e de Razões Estratégicas, utilizados para a descoberta e especificação dos casos de uso do sistema computacional de agendamento de reuniões, são extraídos de Yu (1995). Finalmente, na seção 6.3, são realizadas as considerações finais do capítulo.

6.1 Sistema de Comércio Eletrônico para a Empresa Media Shop

A ***Media Shop*** é uma loja que vende e entrega vários tipos diferentes de itens de mídia, tais como: livros, jornais, revistas, cds de áudio, fitas VHS, DVDs e outros. Os ***clientes*** (remotos ou locais) da *Media Shop* podem usar um catálogo atualizado regularmente que descreve os itens de mídia disponíveis para especificar o seu pedido. A *Media Shop* é abastecida com os últimos lançamentos, pelos ***produtores de mídia***, e itens já cadastrados, pelos ***fornecedores de mídia***. Para aumentar as vendas, a *Media Shop* decidiu implementar um serviço de vendas pela internet. Com a nova configuração o cliente pode pedir itens da *Media Shop* pessoalmente, por telefone, ou através da

internet. Este novo sistema foi chamado de ***Medi@*** e está disponível na *Web* através das facilidades fornecidas pela ***Telecom***. Ele também usa os serviços financeiros fornecidos pelo ***Banco***, que é especializado em transações financeiras on-line. O objetivo básico do novo sistema é permitir que um cliente on-line examine itens no catálogo da internet através do sistema *Medi@* e possa também fazer pedidos de compra através da internet.

O *Medi@* está disponível para qualquer cliente potencial que possua acesso à internet e um navegador *web*. Não há restrições de registro ou procedimentos de identificação para a navegação no catálogo ou consulta de um item do banco de dados. Mesmo sem efetivar nenhuma compra, um visitante anônimo é considerado um cliente on-line do Medi@.

Clientes potenciais podem pesquisar a loja *on-line* tanto através do catálogo *on-line* ou consultando um item do banco de dados. O catálogo agrupa itens da mídia do mesmo tipo em (sub) hierarquias e gêneros (ex.: cds de áudio classificados em pop, rock, jazz, ópera, música clássica, trilha sonora, ...) de forma que o cliente possa consultar apenas as (sub)categorias que interessa a ele.

Um engenho de pesquisa on-line permite aos clientes, com itens particulares em mente, pesquisarem por título, autor/artista/diretor e campos de descrição através de palavras chaves ou de um texto completo. Se o item não estiver disponível no catálogo, o cliente tem a opção de solicitar a *Media Shop* para fazer um pedido do item desejado ao *Fornecedor de Mídia*.

Detalhes sobre os itens da mídia incluem título, categoria da mídia (ex.: livro) e gênero (ex.: ficção científica), autor/artista/diretor, descrição, editora/gravadora, data, custo e algumas figuras (quando disponíveis).

Através de uma análise do ambiente organizacional descrito acima, podemos descrever o Modelo de ***Dependências Estratégicas*** para a loja Media Shop (Castro et al. 2002), conforme apresentado na figura 20.

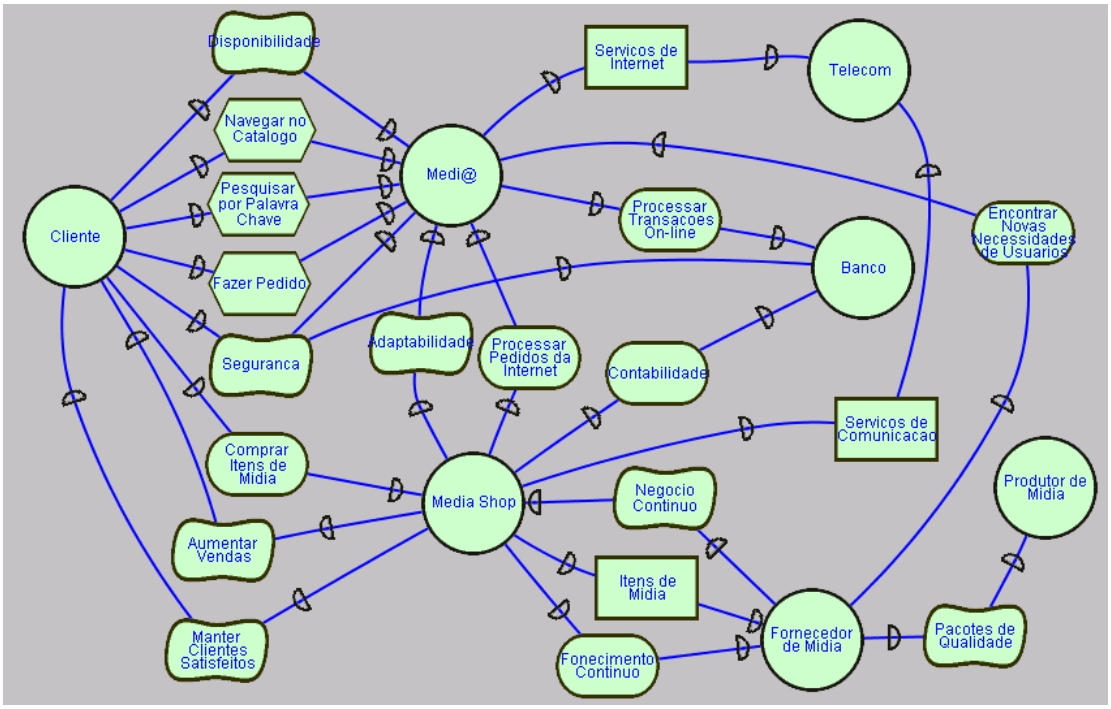

Figura 20. Modelo de Dependências Estratégicas para o sistema Medi@.

Nesta figura, podemos verificar que o *Cliente* depende da loja *Media Shop* para ter seu objetivo *Comprar Itens de Mídia* alcançado. Reciprocamente, a loja *Media Shop* depende do Cliente para *Aumentar as Vendas* e *Manter Clientes Satisfeitos*. Como a meta de *Manter Clientes Satisfeitos* não pode ser definida precisamente, ela é representada como um objetivo-soft. Além disso, *Media Shop* depende do *Fornecedor de Mídia* para prover itens de mídia de forma contínua (dependência de objetivo *Fornecimento Contínuo*) e para adquirir um *Item de Mídia* (dependência de recurso). O *Fornecedor de Mídia* deseja permanecer fornecedor, isto é, manter *Negócio Contínuo*. Por outro lado, *Fornecedor de Mídia* espera que o *Produtor de Mídia* o abasteça com *Pacotes de Qualidade*. Podemos também observar que a loja *Media Shop* depende do

sistema *Medi@* para processar *Pedidos da Internet*. O *Cliente*, por sua vez, depende do *Medi@* para *Fazer Pedidos* através da internet, pesquisar o banco de dados por palavras chaves (*Pesquisar por Palavra Chave*), ou simplesmente *Navegar no Catálogo* on-line. Com respeito à qualidade do serviço, o *Cliente* requer que os serviços de transação sejam seguros e sempre disponíveis (dependências do tipo objetivo-soft *Segurança* e *Disponibilidade*), enquanto a loja *Media Shop* espera que *Medi@* possa ser facilmente adaptado (ex., melhorando o catálogo, evoluindo o banco de dados de itens, atualizando a interface do usuário,...). Finalmente, *Medi@* confia nos *Serviços de Internet* fornecidos pela empresa *Telecom* e depende do *Banco* para processar transações on-line.

Contudo, como visto no capítulo 4, apesar do Modelo de Dependências Estratégicas da figura 20 fornecer uma visão sobre porque processos são estruturados em uma determinada forma, ele não provê mecanismos suficientes para sugerir, explorar e avaliar soluções alternativas. Este tipo de informação pode ser representada através do Modelo de Razões Estratégicas, o qual permite explorar as razões estratégicas associadas com cada ator e suas dependências representadas no Modelo de Dependências Estratégicas. Assim, o modelo de Razões Estratégicas da figura 21 (Castro et al. 2002), permite-nos compreender melhor as razões internas associadas com o sistema computacional ***Medi@*** visando propiciar vendas por internet.

Observando a figura 21, podemos observar as razões estratégicas de ***Medi@.*** No topo (raiz) do refinamento verificamos que a sub-tarefa *Gerenciar Loja da Internet* é inicialmente refinada nos sub-objetivos *Tratar Pedidos da Internet* e *Tratar Pesquisa de Item*, no objetivo-soft *Atrair Novos Clientes* e na sub-tarefa *Produzir Estatísticas*. Para gerenciar pedidos da internet, o sub-objetivo *Tratar Pedidos da Internet* é alcançado através da sub-tarefa colocar item no *Carrinho de Compras,* que é decomposta nas sub-tarefas *Selecionar Item*, *Adicionar Item, Confirmar Compra* e no sub-objetivo *Obter*

Detalhes de Identificação. Essas são as atividades do processo principal que são necessárias para projetar a operacionalização de uma compra através da internet. O sub-objetivo *Obter Detalhes de Identificação* é alcançado tanto através do sub-objetivo *Tratar por Comunicação Clássica* lidando com pedidos *Por Telefone* ou *Por Fax* quanto por *Tratar por Internet* gerenciando pedidos em *Formulário Seguro* ou *Formulário Padrão*. Para permitir o pedido de novos itens não listados no catálogo, *Selecionar Item* é refinado em duas sub-tarefas alternativas (meios), uma dedicada a selecionar itens já catalogados (*Pegar Item Disponível*) e a outra para fazer o pré-pedido de produtos indisponíveis (*Fazer Pré-Pedido de Item Não Disponível*). O sub-objetivo *Tratar Pesquisa de Item* poderia alternativamente ser satisfeito através das tarefas *Consultar Base de Dados* ou *Consultar Catálogo* relacionadas ao desejo de navegação de clientes, isto é, uma pesquisa de itens específicos em mente utilizando mecanismos de pesquisa ou simplesmente, navegando nos produtos catalogados.

6.1.1 Aplicando a Proposta ao Estudo de Caso de Comércio Eletrônico (Sistema Medi@)

Nesta seção, as diretrizes propostas no capítulo 5 são aplicadas ao exemplo de comércio eletrônico descrito na seção anterior. Os modelos organizacionais descritos nas figuras 20 e 21 serão a fonte de informação para descobrir e descrever os casos de uso em UML para o sistema Medi@. Iniciaremos derivando os atores de casos de uso a partir do Modelo de Dependências Estratégicas (figura 20). Em seguida, encontraremos os casos de uso para os atores observando as dependências dos atores no Modelo de Dependências Estratégicas . A partir da observação do Modelo de Razões Estratégicas (figura 21) então, descreveremos o cenário primário (fluxo principal) e secundários (fluxos alternativos) de cada caso de uso descoberto. Por último, uma versão do Diagrama de Caso de Uso em UML para o sistema Medi@ será gerado.

Para facilitar a leitura, será utilizado itálico para referenciar itens oriundos dos diagramas i*, enquanto negrito será usado nos itens dos diagramas de caso de uso.

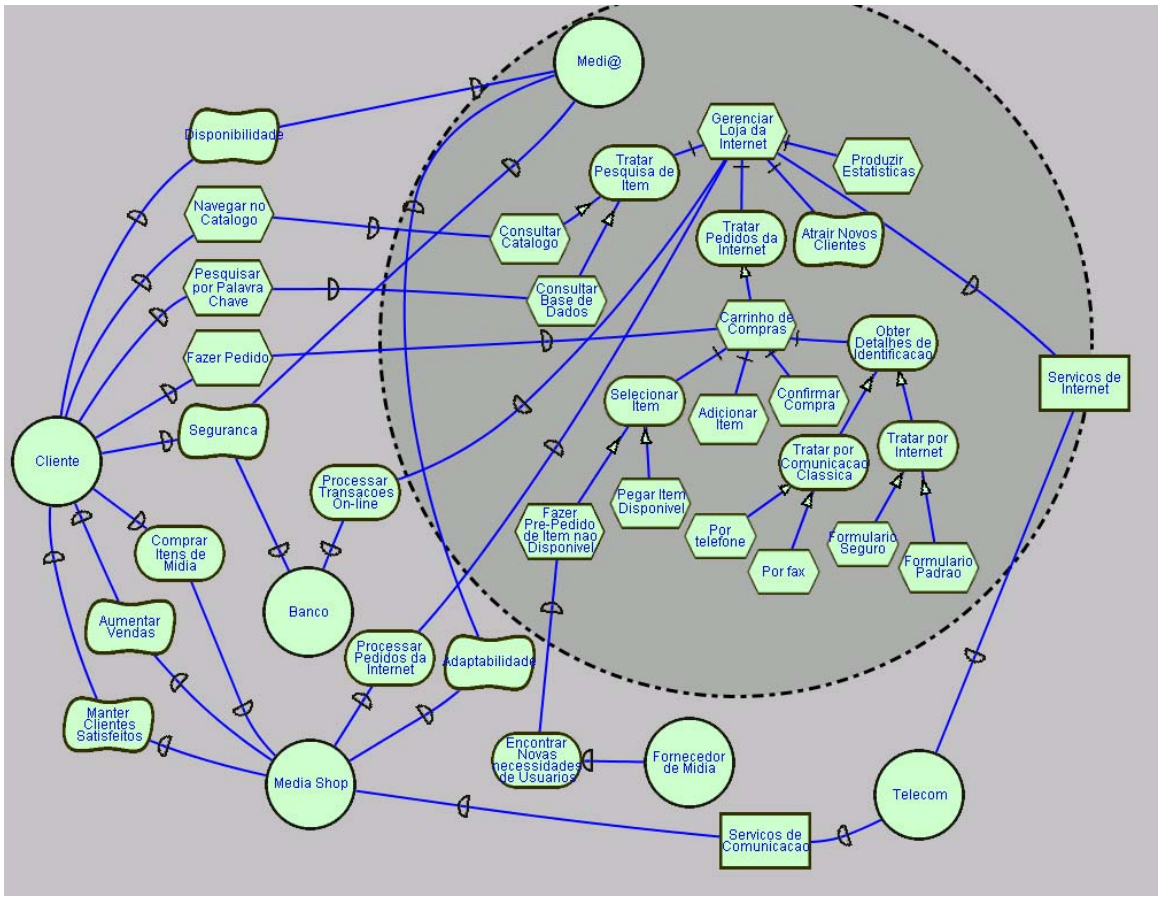

Figura 21. Modelo de Razões Estratégicas para o sistema Medi@.

6.1.1.1 Descobrindo os atores para o sistema Medi@

Inicialmente, a partir da figura 20, podemos extrair os candidatos a atores para o modelo de casos de uso do sistema Medi@. Utilizando as diretrizes apresentadas na seção 5.2.1 (diretrizes 1, 2, 3 e 4), devemos considerar cada um dos atores no Modelo de Dependências Estratégicas para um possível mapeamento para ator em caso de uso. Assim, devemos avaliar os atores *Media Shop, Fornecedor de Mídia, Produtor de Mídia, Banco, Telecom, Cliente* e *Medi@.* Observando mais atentamente, verificamos que o ator *Medi@* fere a diretriz 2. O ator *Medi@* é um sistema computacional, isto é, o

software a ser desenvolvido. Este ator em i* não é externo. Portanto, não pode ser considerado ator em caso de uso. O ator *Produtor de Mídia*, também não pode ser considerado ator em caso de uso, pois não satifaz a diretriz 3, na qual se estabelece que o ator analisado deve possuir pelo menos uma dependência em relação ao sistema computacional pretendido. Isto não ocorre, caracterizando que *Produtor de Mídia* não é relevante do ponto de vista do contexto de interação com o sistema *Medi@* para levar a cabo serviços providos pelo sistema computacional. Os demais atores analisados são considerados apropriados para atores em caso de uso, pois são externos (diretriz 2) ao sistema e suas dependências estratégicas referem-se a aspectos importantes para o desenvolvimento do sistema *Medi@* (diretriz 3). Assim, a lista de atores para o modelo de casos de uso inclui: ***Media Shop, Fornecedor de Mídia, Banco, Telecom e Cliente.*** É importante salientar que outros atores podem futuramente ser descobertos com base na análise e descrição dos casos de uso. Isto pode ocorrer, pois engenheiros de requisitos devem interagir com usuários e clientes para validar os casos de uso gerados e neste processo de validação existe a possibilidade de detectar novos atores. Contudo, o fato de modelos organizacionais serem desenvolvidos integrando os diversos *stakehoders*, contribui positivamente na descoberta e mapeamento dos atores essenciais associados com casos de uso.

O próximo passo (2º passo – seção 5.2.2) na nossa proposta visa descobrir e relacionar casos de uso aos atores descobertos.

6.1.1.2 Descobrindo casos de uso para os atores do sistema Medi@.

Concluída a descoberta dos atores para o sistema *Medi@,* devemos encontrar casos de uso para estes atores com base no Modelo de Dependências Estratégicas da figura 20. Assim, inicialmente seguindo a diretriz 5 e observando a figura 20, podemos gerar a

tabela 4. Esta tabela resume as informações necessárias para realizar o mapeamento de dependências em i* para casos de uso de atores.

Ator	Dependência	Tipo de Dependência	Diretriz a ser usada
Media Shop	Processar Pedidos da internet	Objetivo	(D 6, D 5.1)
Media Shop	Adaptabilidade	Objetivo-Soft	(D 6, D 5.4)
Fornecedor de Mídia	Encontrar Novas Necessidades de Usuários	Objetivo	(D 6, D 5.1)
Banco	Processar Transações On-line	Objetivo	(D 5.1)
Telecom	Serviços de Internet	Recurso	(D 5.3)
Cliente	Navegar no catálogo	Tarefa	(D 6, D5.2)
Cliente	Pesquisar por Palavra Chave	Tarefa	(D 6, D 5.2)
Cliente	Fazer Pedido	Tarefa	(D 6, D 5.2)
Cliente	Disponibilidade	Objetivo-Soft	(D 6, D 5.4)
Cliente	Segurança	Objetivo-Soft	(D 6, D 5.4)

Tabela 4. Informação extraída do Modelo de Dependências Estratégicas para derivar casos de uso do sistema *Medi@*.

- Desta forma, para o ator *Media Shop*, observando-o como *Dependee* (diretriz 5, *Medi@* → *Dependum* → *MediaShop*), verificamos que não existem dependências entre o mesmo e o ator *Medi@*. Contudo, observando este ator como *depender* e o sistema *Medi@* como *dependee* (diretriz 6, *MediaShop* → *Dependum* → *Medi@*)), encontramos a dependência do tipo objetivo *Processar Pedidos da Internet* (ver tabela 4). Esta dependência, como é do tipo objetivo, deve ser analisada sob a ótica da diretriz 5.1. Dessa análise, resulta a criação do caso de uso denominado ***Processar Pedidos da Internet*** para o ator *Media Shop*. A idéia deste caso de uso é reforçada, se considerarmos também, que vários passos são necessários para obter este objetivo, incluindo atividades tais como:

tratar pesquisa de itens de mídia, tratar pedidos na internet e produzir estatísticas. Estas atividades podem ser observadas no Modelo de Razões Estratégicas da figura 21 e mais especificamente, na decomposição da tarefa *Gerenciar Loja da Internet,* a qual é a tarefa encarregada de satisfazer o objetivo *Processar Pedidos da Internet* e conseqüentemente, o caso de uso de mesmo nome originado desta dependência (ver diretriz 6).

- Para encontrar casos de uso para o ator *Fornecedor de Mídia*, as mesmas diretrizes (2º passo da proposta) aplicadas para o ator *Media Shop* podem ser seguidas. Inicialmente, observando o ator *Fornecedor de Mídia* como *dependee* (diretriz 5, *Medi@* → *Dependum* → *Fornecedor de Mídia*), não encontramos nenhuma dependência associada entre o ator e *Medi@*. Contudo, observando o ator como *depender* e o sistema *Medi@* como *dependee* (diretriz 6, *Fornecedor de Mídia* → *Dependum* → *Medi@*), encontramos a dependência do tipo objetivo *Encontrar Novas Necessidades de Usuários* (ver tabela 4). Esta dependência, como é do tipo objetivo, deve ser analisada sob a ótica da diretriz 5.1. Desta análise, resulta a criação do caso de uso denominado ***Encontrar Novas Necessidades de Usuários*** para o ator ***Fornecedor de Mídia***. Note que independente de observar o Modelo de Razões Estratégicas da figura 21 para verificar como *Medi@* lidará com este objetivo, temos optado por criar este caso de uso. A observação do Modelo de Razões Estratégicas , contudo, indica que o objetivo *Encontrar Novas Necessidades de Usuários* é satisfeito internamente por *Medi@,* através da tarefa *Fazer Pré-Pedido de Item não Disponível*. Esta tarefa pode ser o elemento do início da investigação, de quais passos poderíamos incluir no caso de uso ***Encontrar Novas Necessidades de Usuários,*** visando

encontrar possíveis novas necessidades de usuários para o ***Fornecedor de Mídia***. Assim, este caso de uso conterá os vários passos necessários para detectar novas necessidades de usuários com base, principalmente, em solicitações de produtos não disponíveis. Por exemplo, o fornecedor poderia inicialmente verificar quais são os pré-pedidos de itens não disponíveis, em seguida poderia verificar qual é o tipo de item não disponível e/ou não existente mais solicitado, a quais regiões geográficas pertencem os clientes que realizaram esses pré-pedidos, realizar uma consulta aos clientes via questionário (e-mail), procurando identificar novas necessidades e por fim, possivelmente otimizar o processo de fornecimento desses itens (mídias) e atender melhor as novas necessidades de clientes.

- Para o ator *Banco*, observando-o como *dependee* (diretriz 5, *Medi@* → *Dependum* → *Banco*) encontramos a dependência do tipo objetivo *Processar Transações On-line* (ver tabela 4). Esta dependência pode ser mapeada para o caso de uso denominado ***Processar Transações On-line***, o qual conterá os vários passos na interação entre ***Banco*** e *Medi@* para lidar com as transações financeiras on-line.
- Para o ator *Telecom*, observando-o como *dependee* (diretriz 5, *Medi@* → *Dependum* → *Telecom*), encontramos a dependência do tipo recurso *Serviços de Internet*. Seguindo a diretriz 5.3, podemos concluir que o principal objetivo associado com a necessidade de disponibilidade do recurso serviços de internet é prover o sistema *Medi@* com serviços de internet necessários para realização de vendas on-line. Consideramos que para prover estes serviços, seriam necessários vários passos de interação entre o sistema *Medi@* e o ator *Telecom*, os quais

seriam descritos no caso de uso denominado de ***Serviços de Internet*** associado com o ator ***Telecom***.

- Para o ator *Cliente* podemos indicar alguns casos de uso originados dos seus relacionamentos de dependências. Inicialmente, observando este ator como *dependee* (diretriz 5, *Medi@* → *Dependum* → *Cliente*), não encontramos dependências entre o ator e o sistema Medi@. Contudo, observando Cliente como *depender* e Medi@ como *dependee* (diretriz 6, *Cliente* → *Dependum* → *Medi@*), podemos verificar a existência de três dependências do tipo tarefa que podem originar casos de uso: *Navegar no Catálogo*, *Pesquisar por Palavra Chave* e *Fazer Pedido* (ver tabela 4). Estas dependências devem ser analisadas com o auxílio da diretriz 5.2. A dependência Navegar no Catálogo pode ser mapeada para o caso de uso ***Navegar no Catálogo***, o qual conterá os vários passos de interação requeridos para navegar no catálogo na internet. A dependência *Pesquisar por Palavra Chave* pode ser mapeada para o caso de uso ***Pesquisar por Palavra Chave***, o qual conterá os vários passos de interação requeridos entre ***Cliente*** e *Medi@* para realizar uma pesquisa utilizando uma palavra chave. Finalmente, a dependência *Fazer Pedido* pode ser mapeada para o caso de uso ***Fazer Pedido***, o qual conterá os vários passos de interação entre ***Cliente*** e *Medi@* para fazer um pedido através da internet.

  É importante observar que a descrição do cenário primário (fluxo principal) de cada um dos casos de uso descobertos para os atores do sistema, pode dar origem a outros casos de uso. Isto exige uma análise posterior à descrição dos casos de uso, o que será realizado seguindo-se a diretriz 9. Neste momento, o que temos, são os casos de uso essenciais para o sistema Medi@.

- Em relação aos objetivos-soft, que representam os requisitos não-funcionais do sistema *Medi@*, podemos estabelecer a seguinte análise com base na diretriz 5.4. Segundo esta diretriz, podemos mapear os objetivos-soft com os atores (ver tabela 4) para requisitos não-funcionais do sistema *Medi@*. Observando mais atentamente a tabela 4, verificamos que o ator *Media Shop* tem associado o objetivo-soft *Adaptabilidade* em relação ao sistema *Medi@,* estabelecendo a expectativa de que o sistema possa ser facilmente adaptado (ex., melhorando o catálogo, evoluindo o banco de dados de itens, atualizando a interface do usuário,...). Da mesma forma, verificamos também que clientes possuem duas dependências do tipo objetivo-soft em relação a *Medi@* (ver tabela 4): *Segurança e Disponibilidade*, estabelecendo a expectativa de clientes que esperam que os serviços de transação providos pelo sistema *Medi@* sejam seguros e estejam disponíveis. Desta forma, estes três objetivos-soft são definidos como requisitos não funcionais para o sistema *Medi@*. Uma descrição complementar em relação ao modelo de casos de uso pode incluir a descrição destes requisitos. É importante salientar, que conforme descrito na diretriz 8.1, objetivos-soft modelados no Modelo de Razões Estratégicas da figura 21, serão posteriormente analisados (próxima seção), em um possível mapeamento para requisitos não-funcionais relacionados a casos de uso específicos já descobertos para o sistema *Medi@*.
- Depois de derivados os casos de uso para o sistema *Medi@*, podemos então seguindo a diretriz 7, classificar cada caso de uso e seu objetivo associado, como mostrado na tabela 5.

Assim, fazendo uso das diretrizes propostas na seção 5.2.2, pudemos derivar casos de uso para o sistema *Medi@* observando as dependências no Modelo de Dependências Estratégicas da figura 20. Os casos de uso descobertos incluem: **Processar Pedidos da Internet** para o ator **Media Shop**, **Encontrar Novas Necessidades de Usuários** para o ator **Fornecedor de Mídia**, **Processar Transações On-line** para o ator **Banco**, **Serviços de Internet** para o ator **Telecom** e os casos de uso **Navegar no Catálogo**, **Pesquisar por Palavra Chave** e **Fazer Pedido** para o ator **Cliente**. Continuando o processo de derivação, podemos agora iniciar a descrição/especificação dos cenários primário e secundários (fluxos principal e alternativos) dos casos de uso descobertos. Na próxima seção realizaremos este passo.

Ator	Objetivo do Caso de Uso	Classificação do Objetivo
Media Shop	Processar Pedidos da Internet	Objetivo de Contexto
Fornecedor de Mídia	Encontrar Novas Necessidades de Usuários	Objetivo de Usuário
Banco	Processar Transações On-line	Objetivo de Contexto
Telecom	Serviços de Internet	Objetivo de Usuário
Cliente	Navegar no Catálogo	Subfunção
Cliente	Pesquisar por Palavra Chave	Subfunção
Cliente	Fazer Pedido	Objetivo de Usuário

Tabela 5. Classificação de Objetivos para o sistema Medi@.

6.1.1.3 Descrevendo os casos de uso do sistema Medi@.

Conforme diretriz 8, cabe neste passo observar o modelo de Razões Estratégicas apresentado na figura 21, buscando informações que possam nos auxiliar na descrição dos casos de uso para o sistema Medi@.

Por exemplo, vejamos a possibilidade de descrever o caso de uso **Fazer Pedido** para o ator Cliente (ver tabela 5) a partir da observação do Modelo de Razões Estratégicas da figura 21. Este caso de uso representa o uso do sistema Medi@ pelo Cliente para

fazer um pedido. Observando a figura 21 mais detalhadamente, verificamos que o sistema Medi@ satisfaz a dependência *Fazer Pedido* (*já mapeada para o caso de uso* ***Fazer Pedido***) através da tarefa colocar item no *Carrinho de Compras* e suas sub-tarefas *Adicionar Item* e *Confirmar Compra* e seus sub-objetivos *Selecionar Item* e *Obter Detalhes de Identificação.* Estes sub-componentes (conforme descrito na diretriz 8.1) podem ser mapeados para o cenário primário (fluxo principal) do caso de uso ***Fazer Pedido.*** Salientamos, que utilizaremos na nossa proposta, o *template* de especificação de caso de uso proposto por Cockburn (2000) (ver figura 5 na seção 3.2), com pequenas adaptações, visando uma versão mais simplificada que se adapte melhor ao nosso trabalho. Vejamos então, como podemos descrever o caso de uso ***Fazer Pedido***:

Caso de Uso: 1 Fazer Pedido

Objetivo no Contexto: Cliente deseja fazer um pedido por internet e espera que seu pedido seja aceito.
Escopo: Sistema Medi@
Nível: Objetivo de usuário
Pré-condições:
Condicao Final de Sucesso: Item Colocado no Carrinho de Compras, Pedido Feito
Ator: Cliente

CENÁRIO PRINCIPAL
1. O caso de uso inicia com o Cliente selecionando um item disponível.
2. O cliente adiciona o item ao carrinho de compras.
3. O cliente faz a confirmação do pedido de compra.
4. O sistema obtém detalhes de identificação do cliente através da internet e encerra o pedido.

EXTENSÕES
1a. O item que Cliente seleciona não está disponível:
    1a1. Cliente Faz Pré-Pedido de Item não disponível.
4a. O cliente não deseja fornecer detalhes de identificação através da internet:
    4a1. O sistema obtém detalhes de identificação do cliente através de comunicação clássica.

INFORMAÇÃO RELACIONADA
Caso de Uso Pai:
Casos de Uso Subordinados:

Figura 22. Especificação do caso de uso Fazer Pedido

Assim, seguindo a diretriz 8, procedemos a observação do Modelo de Razões Estratégicas da figura 21, buscando quais elementos estão envolvidos na obtenção da tarefa *Fazer Pedido* pelo ator *Medi@.* Este ator tem a responsabilidade de levar a cabo esta tarefa que tem sido anteriormente (ver tabela 5), mapeada para caso de uso. Assim, observamos que os passos 1, 2, 3 e 4 do caso de uso da figura 22, são extraídos da ligação de decomposição da tarefa colocar item no *Carrinho de Compras,* cuja função é satisfazer a dependência do tipo tarefa *Fazer Pedido* (ver figura 21). O passo 1, neste caso de uso, é originado da sub-tarefa *Selecionar Item*, estabelecendo que o *Cliente* inicialmente precisa selecionar um item. Podemos também observar que, *Selecionar Item* pode ser obtido (via ligação meio-fim), através de duas tarefas alternativas, uma dedicada a selecionar itens catalogados e a outra; a fazer um pré-pedido de um item não disponível. Assim, *Cliente* pode selecionar um item catalogado (escolhido como caminho normal no cenário e por isso descrito no passo 1) ou fazer um pré-pedido de um item não disponível (definido como passo alternativo ao passo 1 e descrito como 1.a.1 na seção de extensões, considerando que esta ação é geralmente uma exceção). O passo 2 foi derivado da observação da sub-tarefa *Adicionar Item*, estabelecendo que o *Cliente* necessita adicionar o item selecionado ao carrinho de compras (veja figura 21). O passo 3 é originado da sub-tarefa *Confirmar Compra*. Finalmente, o passo 4 é oriundo da observação do sub-objetivo *Obter Detalhes de Identificação*, estabelecendo que o *Cliente* necessita prover o sistema com seus detalhes de identificação para que o pedido seja realizado com sucesso. Esta atividade pode ser alternativamente obtida (via ligação meio-fim), através de dois sub-objetivos: *Tratar por Internet* (escolhido com a ação normal no cenário primário e assim descrito no passo 4) ou *Tratar por Comunicação Clássica* (definido como item 4.a.1 na seção de extensões) (ver diretriz 8.2). Observamos também que, o caso de uso não possui pré-condições por não

existirem dependências da tarefa colocar item no *Carrinho de Compras* em relação a outros atores do sistema (ver diretriz 8.3). No entanto, podemos definir seguindo a diretriz 8.3, duas *pós-condições finais de sucesso* do caso de uso. A primeira é derivada da tarefa colocar item no *Carrinho de Compras* que satisfaz a dependência *Fazer Pedido* mapeada para caso de uso. Esta tarefa é escrita no passado (*item colocado no Carrinho de Compras*) para demonstrar que é uma condição obtida após a execução com sucesso do caso de uso. A outra *pós-condição de sucesso* descrita é *Pedido Feito* derivada do próprio objetivo obtido com sucesso do caso de uso.

Visando melhorar a especificação do caso de uso Fazer Pedido, observamos que seguindo a diretriz 9, podemos evoluir na descoberta de novos casos de uso a partir da análise dos passos da especificação do caso de uso gerada até o momento. Assim, observando o caso de uso Fazer Pedido (figura 22), podemos analisar os seus passos conforme segue:

- Passo 1 do caso de uso: "O caso de uso inicia com o Cliente selecionando um item disponível". O objetivo desta sentença, respondendo à pergunta da diretriz 9.1, é "Selecionar um Item Disponível". Observando a diretriz 9.2, deve-se investigar quais são os passos necessários (sub-objetivos) para se obter esse objetivo. Tipicamente, em um sistema de comércio eletrônico tal como o sistema Medi@, o fato de se selecionar um item disponível implica em uma ação simples de clicar em cima de um item já pesquisado. Assim, não se justifica e não cabe aqui, criar um novo caso de uso para este passo.
- Passo 2 do caso de uso: "O cliente adiciona o item ao carrinho de compras". O objetivo desta sentença, respondendo à pergunta da diretriz 9.1, é "Adicionar Item ao Carrinho de Compras". Observando a diretriz

9.2, deve-se investigar quais são os passos necessários (sub-objetivos) para se obter esse objetivo. Tipicamente, em um sistema de comércio eletrônico tal como o sistema Medi@, o fato de se adicionar um item ao carrinho de compras implica em uma ação realizada em um simples clique. Assim, não se justifica e não cabe aqui, criar um novo caso de uso para este passo.

- Passo 3 do caso de uso: "O cliente faz a confirmação do pedido". O objetivo desta sentença, respondendo à pergunta da diretriz 9.1, é "Confirmar Pedido". Observando a diretriz 9.2, deve-se investigar quais são os passos necessários (sub-objetivos) para se obter esse objetivo. Tipicamente, em um sistema de comércio eletrônico tal como o sistema Medi@, o fato de se confirmar um pedido implica em uma ação bastante simples que requer, antes de tudo, uma tomada de decisão do cliente para confirmar o pedido. Assim, não se justifica e não cabe aqui, criar um novo caso de uso para este passo.
- Passo 4 do caso de uso: "O sistema obtém detalhes de identificação do cliente através da internet e encerra o pedido". O objetivo desta sentença, respondendo à pergunta da diretriz 9.1, é "Obter detalhes de identificação via internet". Observando a diretriz 9.2, deve-se investigar quais são os passos necessários (sub-objetivos) para se obter esse objetivo. Tipicamente, em um sistema de comércio eletrônico tal como o sistema Medi@, o fato do cliente fornecer detalhes de identificação pode incluir passos tais como: se é um novo cliente, algumas informações pessoais adicionais podem ser solicitadas como: endereço, e-mail, telefone, etc (o que não ocorre para clientes já cadastrados); o cliente deve selecionar um

meio de envio dos itens (correio normal, sedex, etc); pode-se escolher também uma forma de pagamento a qual é verificada on-line pelo sistema e se houver problemas deve-se possibilitar ao cliente cancelar o pedido; além disso, normalmente o cliente deve cadastrar uma "senha" e "conta" para poder acompanhar ou mesmo cancelar o pedido de acordo com certas restrições. Enfim, verificamos que vários passos de interação entre Cliente e o sistema Medi@ seriam necessários para poder obter o objetivo do passo 4 do caso de uso. Desta forma, seguindo a diretriz 9.3, um caso de uso denominado de ***Obter Detalhes de Identificação via Internet*** poderia ser gerado para incluir os passos apresentados acima. Esta relação deve ser expressa via mecanismo de inclusão <<*include*>> adicionado ao passo 4, conforme segue:

- **Passo 4:** "O sistema obtém detalhes de identificação do cliente através da internet e encerra o pedido. (O caso de uso ***Obter Detalhes de Identificação via Internet*** é incluído <<*include*>> neste passo)".

- Passo 1.a do caso de uso (extensão): "Cliente Faz Pré-Pedido de Item não Disponível". O objetivo desta sentença, respondendo à pergunta da diretriz 9.1, é "Fazer Pré-Pedido de Item não Disponível". Observando a diretriz 9.2, deve-se investigar quais são os passos necessários (sub-objetivos) para se obter esse objetivo. Tipicamente, fazer um pré-pedido de um item não disponível em um sistema de comércio eletrônico pode incluir passos tais como: selecionar ou especificar o item a ser pré-pedido, adicionar o item e a quantidade ao pré-pedido, confirmar o pré-pedido, fornecer detalhes pessoais de identificação tais como: endereço,

e-mail, telefone, etc, possivelmente criar um *login* e senha para poder acompanhar o progresso do pré-pedido, etc. Enfim, verificamos que vários passos de interação entre Cliente e o sistema Medi@ seriam necessários para poder obter o objetivo do passo 1.a (seção de extensões) do caso de uso. Desta forma, seguindo a diretriz 9.3, um caso de uso denominado de ***Fazer Pré-Pedido de Item Não Disponível*** poderia ser gerado para incluir os passos apresentados acima. Esta relação deve ser descrita no passo de extensão do use case, conforme proposto no *template* da figura 5. Desta forma, a ação no passo 1a1 "Cliente Faz Pré-Pedido de Item não Disponível" é substituída pela declaração do novo sub.caso de uso chamado "***Fazer Pré-Pedido de Item não Disponível***". Assim, o passo 1a é descrito conforme segue:

Passo 1a: O item que Cliente seleciona não está disponível:

1a1. ***Fazer Pré-Pedido de Item Não Disponível***

- Passo 4.a do caso de uso (extensão): "O sistema obtém detalhes de identificação do cliente através de comunicação clássica". O objetivo desta sentença, respondendo à pergunta da diretriz 9.1, é "Obter Detalhes de Identificação via Comunicação Clássica". Observando a diretriz 9.2, deve-se investigar quais são os passos necessários (sub-objetivos) para se obter esse objetivo. Tipicamente, a obtenção de detalhes pessoais de identificação utilizando os métodos tradicionais, tais como: telefone ou fax, normalmente envolve um processo manual de contato entre o cliente e um atendente, sendo que os dados de identificação podem ser preenchidos, posteriormente, no sistema pelo atendente. Assim, consideramos para o sistema Medi@ que a comunicação via meio

tradicional (telefone ou fax) é uma interação que não justifica a criação de um novo caso de uso.

Desta forma, após esta análise mais aprofundada dos passos da especificação do caso de uso Fazer Pedido (diretriz 9), a seguinte especificação pode ser gerada, a qual está atualizada com as modificações resultantes desta análise adicional.

Caso de Uso: 1 Fazer Pedido

Objetivo no Contexto: Cliente deseja fazer um pedido por internet e espera que seu pedido seja aceito.
Escopo: Sistema Medi@
Nível: Objetivo de usuário
Precondições:
Condicao Final de Sucesso: Itens Colocados no Carrinho de Compras, Pedido Feito
Ator: Cliente

CENÁRIO PRINCIPAL
1. O caso de uso inicia com o Cliente selecionando um item disponível.
2. O cliente adiciona o item ao carrinho de compras.
3. O cliente faz a confirmação do pedido de compra.
4. O sistema obtém detalhes de identificação do cliente através da internet e encerra o pedido. (O caso de uso ***Obter Detalhes de Identificação via Internet*** é incluído <<include>> neste passo).

EXTENSÕES
1a. O item que Cliente seleciona não está disponível:
1a1. ***Fazer Pré-Pedido de Item Não Disponível***
4a. O cliente não deseja fornecer detalhes de identificação através da internet:
4a1. O sistema obtém detalhes de identificação do cliente através de comunicação clássica.

INFORMAÇÃO RELACIONADA
Caso de Uso Pai:
Casos de Uso Subordinados:
Obter detalhes de Identificação via Internet
Fazer Pré-Pedido de Item Não Disponível

Figura 23. Especificação melhorada do caso de uso Fazer Pedido

De forma análoga à análise do caso de uso ***Fazer Pedido,*** podemos também descrever o caso de uso ***Processar Pedidos na Internet*** para o ator ***Fornecedor de Mídia*****,** usando inicialmente a diretriz 8. Este caso de uso deveria conter todos os passos

necessários para processar pedidos realizados pela internet. Observando a figura 21, verificamos que este objetivo é satisfeito pelo sistema *Medi@* através da tarefa *Gerenciar Loja da Internet* e seus sub-objetivos *Tratar Pesquisa de Item* e *Tratar Pedidos da Internet* bem como pela sub-tarefa *Produzir Estatísticas* e pelo objetivo-soft *Atrair Novos Clientes* (ver figura 21). Assim, para o caso de uso ***Processar Pedidos da Internet*** teríamos a seguinte especificação:

Caso de Uso: 2 Processar Pedidos da Internet

Objetivo no Contexto: Media Shop deseja processar os pedidos realizados pelos clientes via internet.
Escopo: Sistema Medi@
Nível: Objetivo de contexto
Precondições: Serviços de Internet Providos, Transações On-line Processadas
Condição Final de Sucesso: Loja da Internet Gerenciada, Pedidos da Internet Processados
Ator: Media Shop

CENÁRIO PRINCIPAL
1. O caso de uso inicia quando Media Shop requer que a pesquisa de itens seja tratada via consulta à base de dados. (O caso de uso ***Pesquisar por Palavra Chave*** é incluído <<*include*>> neste passo)
2. O sistema trata os pedidos realizados pela internet via carrinho de compras. (O caso de uso ***Fazer Pedido*** é incluído <<*include*>> neste passo)
3. O sistema produz estatísticas para Media Shop.

EXTENSÕES
1a.A pesquisa de item é realizada via consulta ao catálogo:
1a1. ***Navegar no Catálogo***

INFORMAÇÃO RELACIONADA
Caso de Uso Pai:
Casos de Uso Subordinados:
Pesquisar por Palavra Chave
Fazer Pedido
Navegar no Catálogo

Requisitos Especiais: Atrair Novos Clientes

Figura 24. Especificação do caso de uso Processar Pedidos da Internet

Os passos 1, 2 e 3 do caso de uso acima tem como origem as decomposições da sub-tarefa Gerenciar Loja da Internet, a qual tem e responsabilidade de satisfazer a dependência do tipo objetivo *Processar Pedidos da Internet* (ver figura 21). O passo 1 tem origem no sub-objetivo *Tratar Pesquisa de Item*, estabelecendo que *Media Shop* requer que a pesquisa de itens seja tratada corretamente. Observamos também que, tratar Pesquisa de Item poderia alternativamente ser satisfeita (via ligação meio fim), através das sub-tarefas *Consultar Base de Dados* ou *Consultar Catálogo*, de forma que *Cliente* poderia pesquisar por um item específico utilizando funções de pesquisa do sistema ou navegar sobre os itens já catalogados. Assim, a loja *Media Shop* requer que uma pesquisa de item seja tratada através de *Consultar Base de Dados* (escolhida com a ação normal no cenário primário e assim descrita no passo 1) ou alternativamente, seja tratada via *Consultar Catálogo* (definida como passo alternativo 1.a.1 na seção de extensões). Note também que, a sub-tarefa *Consultar Base de Dados* satisfaz a dependência do tipo tarefa *Pesquisar por Palavra Chave*, mapeada anteriormente (ver tabela 5) para o caso de uso ***Pesquisar Por Palavra Chave***. Com base nesta observação, o caso de uso ***Pesquisar Por Palavra Chave*** é incluído <<*include*>> no passo 1 porque este representa os passos necessários (ações) para realizar uma pesquisa através da consulta à base de dados. Alternativamente, como a sub-tarefa *Consultar Catálogo* satisfaz a dependência do tipo tarefa *Navegar no Catálogo* (já mapeada para o caso de uso ***Navegar no Catálogo –*** ver tabela 5), temos descrito no passo 1.a.1 (seção de extensões) o caso de uso ***Navegar no Catálogo***, considerando que este caso de uso representa os passos necessários para realizar uma pesquisa de item via consulta ao catálogo on-line.

O passo 2 é derivado do sub-objetivo *Tratar Pedidos da Internet*. Este sub-objetivo (ver figura 21), estabelece a necessidade de tratar os pedidos realizados pela internet.

Contudo, verificamos que esta atividade é obtida (via ligação meio-fim), através da sub-tarefa colocar item no *Carrinho de Compras*. Assim, o passo 2, mostra o caminho normal no cenário primário tratando pedidos da internet via carrinho de compras. No entanto, de forma análoga ao passo 1, verificamos que a sub-tarefa colocar item no *Carrinho de Compras* é responsável por satisfazer a dependência do tipo tarefa *Fazer Pedido* (ver tabela 5), já mapeada para o caso de uso ***Fazer Pedido***. Assim, com base nesta observação, incluímos via mecanismo <<*include*>> no passo 2, o caso de uso ***Fazer Pedido***.

O passo 3 tem origem na sub-tarefa *Produzir Estatísticas*. Este passo representa as ações (interações entre *Media Shop* e o sistema *Medi@*) para produzir estatísticas que possam ser usadas por *Media Shop* para melhorar o negócio de comércio eletrônico.

O último sub-componente da sub-tarefa *Gerenciar Loja da Internet* é o objetivo-soft *Atrair Novos Clientes*, o qual conforme diretriz 8.1, tem sido mapeado para requisito não-funcional do caso de uso ***Processar Pedidos da Internet***. Este requisito é descrito na seção Requisitos Especiais na especificação do caso de uso.

Observamos também que, o caso de uso acima possui duas pré-condições: ***Serviços de Internet Providos e Transações On-line Processadas***. Estas pré-condições, conforme diretriz 8.3, são derivadas respectivamente da dependência do tipo recurso *Serviços de Internet* entre a tarefa *Gerenciar Loja da Internet* (que satisfaz a dependência *Processar Pedidos da Internet* já mapeada para caso de uso) e o ator Telecom e a dependência do tipo objetivo *Processar Transações On-line* entre a mesma tarefa e o ator Banco. Podemos também definir seguindo a diretriz 8.3, duas *pós-condições finais de sucesso* do caso de uso. A primeira é derivada da tarefa *Gerenciar Loja da Internet* que satisfaz a dependência *Processar Pedidos da Internet* mapeada para caso de uso. Esta tarefa é escrita no passado (*Loja da Internet Gerenciada*) para

demonstrar que é uma condição obtida após a execução com sucesso do caso de uso. A outra pós-condição final de sucesso descrita é *Pedidos da Internet Processados* derivada do próprio objetivo obtido com sucesso do caso de uso.

Da mesma forma que realizamos uma análise adicional do caso de uso ***Fazer Pedido***, podemos também melhorar a especificação do caso de uso ***Processar Pedidos da Internet*** seguindo a diretriz 9. Com base nessa diretriz, podemos evoluir na descoberta de novos casos de uso a partir da análise dos passos da especificação do caso de uso ***Processar Pedidos da Internet*** gerada até o momento. Assim, observando o caso de uso ***Processar Pedidos da Internet*** (figura 24), podemos analisar seus passos conforme segue:

- Inicialmente verificamos que os passos 1 e 2 do cenário primário (fluxo principal) e o passo 1.a.1 da seção de extensões do caso de uso ***Processar Pedidos da Internet***, já representam passos que incluem casos de uso. Assim, não cabe para este passos aplicar a diretriz 9.
- Já o passo 3, necessita ser investigado. Este passo/sentença é assim expresso no caso de uso (ver figura 24): "O sistema produz estatísticas para Media Shop". O objetivo desta sentença, respondendo à pergunta da diretriz 9.1, é "Produzir Estatísticas". Observando a diretriz 9.2, deve-se investigar quais são os passos necessários (sub-objetivos) para se obter esse objetivo. Tipicamente, produzir estatísticas de pedidos para *Media Shop* poderia envolver entre outros aspectos: definir o tipo de estatística de pedidos desejada podendo ser, por exemplo, pedidos de um item específico, pedidos de uma categoria de itens específica, ou pedidos em geral; definir o período (dias, semanas, meses, ano, etc), sobre o qual quer se produzir dados estatísticos; estabelecer sobre qual região

geográfica, por exemplo, quer se produzir estatísticas sobre pedidos. Enfim, pode-se definir uma série de sub-objetivos (passos) que seriam necessários para poder alcançar o objetivo do passo 3 "Produzir Estatísticas" do caso de uso observado. Desta forma, seguindo a diretriz 9.3, um caso de uso denominado de ***Produzir Estatísticas*** poderia ser gerado para incluir os passos apresentados acima. Esta relação deve ser expressa via mecanismo de inclusão <<*include*>> adicionado ao passo 3, conforme segue:

- Passo 3: "O sistema produz estatísticas para Media Shop". (O caso de uso ***Produzir Estatísticas*** é incluído <<*include*>> neste passo).

Desta forma, após esta análise mais aprofundada dos passos da especificação do caso de uso ***Processar Pedidos da Internet*** (diretriz 9), a especificação para este caso de uso pode ser atualizada com as modificações resultantes desta análise adicional, conforme apresentado na figura 25.

Assim, após termos descobertos os atores, bem como alguns casos de uso e suas especificações para o sistema Medi@, podemos conforme diretriz 10, gerar uma versão do Diagrama de Casos de Uso em UML para Medi@. A figura 26 apresenta este diagrama. Salientamos que este diagrama, bem como as especificações de casos de uso realizadas, podem ser modificadas ou complementadas à medida que novos relacionamentos são encontrados. A nossa proposta visa encontrar os casos de uso essenciais do sistema e, a partir deste ponto, poder evoluir em relação à definição funcional completa de um sistema computacional.

Caso de Uso: 2 Processar Pedidos da Internet

Objetivo no Contexto: Media Shop deseja processar os pedidos realizados pelos clientes via internet.
Escopo: Sistema Medi@
Nível: Objetivo de contexto
Precondições: Serviços de Internet Providos, Transações On-line Processadas
Condição Final de Sucesso: Loja da Internet Gerenciada, Pedidos Processados
Ator: Media Shop

CENÁRIO PRINCIPAL
1. O caso de uso inicia quando Media Shop requer que a pesquisa de itens seja tratada via consulta à base de dados. (O caso de uso ***Pesquisar por Palavra Chave*** é incluído <<*include*>> neste passo)
2. O sistema trata os pedidos realizados pela internet via carrinho de compras. (O caso de uso ***Fazer Pedido*** é incluído <<*include*>> neste passo)
3. O sistema produz estatísticas para Media Shop. (O caso de uso ***Produzir Estatísticas*** é incluído <<include>> neste passo)

EXTENSÕES
1a.A pesquisa de item é realizada via consulta ao catálogo:
1a1. ***Navegar no Catálogo***

INFORMAÇÃO RELACIONADA
Caso de Uso Pai:
Casos de Uso Subordinados:
Navegar no Catálogo
Pesquisar por Palavra Chave
Produzir Estatísticas
Fazer Pedido

Requisitos Especiais: Atrair Novos Clientes

Figura 25. Especificação melhorada do caso de uso Processar Pedidos da Internet

Podemos também verificar que, com as informações existentes no Modelo de Razões Estratégicas da figura 21, foi possível gerar as especificações para estes dois casos de uso. Salientamos que, quanto mais razões estratégicas forem modeladas nos Modelos SR, maior será a possibilidade de se aprofundar na descrição dos casos de uso descobertos. O que queremos mostrar, com este exemplo, é viabilidade de derivar casos

de uso e especificações de uma forma controlada e gerenciável facilitando o trabalho de engenheiros de requisitos e desenvolvedores.

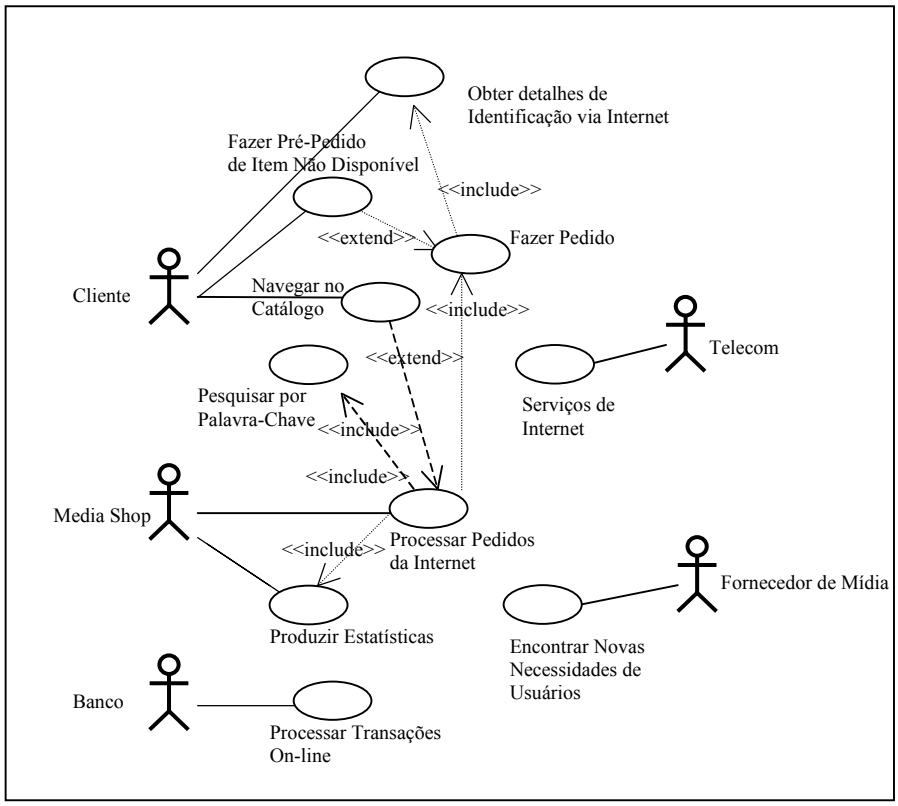

Figura 26. Diagrama de Casos de Uso para o sistema Medi@.

6.2 Problema de Agendamento de Reuniões

O problema de agendamemto de reuniões é um problema bastante conhecido na literatura da Engenharia de software/requisitos. Este problema já foi inicialmente introduzido na seção 4.1, mostrando a necessidade de uma organização realizar agendamentos de reuniões com um menor esforço e tempo possíveis. Na seção 4.1, apresentamos o Modelo de Dependências Estratégicas (figura 13), e o Modelo de Razões Estratégicas (figura 14), ambos extraídos de Yu (1995). Estes modelos representam respectivamente, as dependências externas entre os atores organizacionais,

bem como as razões estratégicas internas destes atores, em relação a processos da organização.

6.2.1 Aplicando a Proposta ao Estudo de Caso de Agendamento de Reuniões (Sistema Agendador de Reunião)

Assim, o nosso objetivo agora é aplicar as diretrizes propostas no capítulo 5 ao problema de agendamento de reuniões. Os modelos organizacionais, descritos nas figuras 13 e 14, serão a fonte de informação para descobrir e descrever os casos de uso em UML para o sistema Agendador de Reunião. Iniciaremos derivando os atores de casos de uso a partir do Modelo de Dependências Estratégicas (figura 13). Em seguida, encontraremos os casos de uso para os atores, observando as dependências dos atores no modelo de Dependências Estratégicas . A partir da observação do Modelo de Razões Estratégicas (figura 14), então descreveremos o cenário primário (fluxo principal) e secundários (fluxos alternativos), de cada caso de uso descoberto. Por último, uma versão do diagrama de caso de uso em UML para o sistema Agendador de Reunião será gerado. Vale salientar que uma versão preliminar do mapeamento foi apresentada em (Santander et al. 2002).

6.2.1.1 Descobrindo casos de uso para os atores do sistema Agendador de Reunião

Inicialmente, a partir da figura 13, podemos extrair os candidatos a atores para o modelo de casos de uso do sistema *Agendador de Reunião*. Utilizando as diretrizes apresentadas na seção 5.2.1 (diretrizes 1, 2, 3 e 4), devemos observar cada um dos atores no Modelo de Dependências Estratégicas para um possível mapeamento para ator em caso de uso. Assim, devemos avaliar os atores *Iniciador de Reunião***,** *Participante de Reunião*, *Participante Importante* e *Agendador de Reunião*. Observando mais atentamente, verificamos que o ator *Agendador de Reunião* não

satisfaz a diretriz 2. O ator *Agendador de Reunião* é um sistema computacional, isto é, o software a ser desenvolvido. Este ator em i* não é externo, e portanto, não pode ser considerado ator em caso de uso. Os demais atores analisados são considerados apropriados para atores em caso de uso, pois são externos (diretriz 2) ao sistema e suas dependências estratégicas referem-se a aspectos importantes para o desenvolvimento do sistema *Agendador de Reunião* (diretriz 3). Assim, a lista de atores para o modelo de casos de uso inclui: ***Iniciador de Reunião*, *Participante de Reunião e Participante Importante*.** Observando melhor a figura 13, verificamos que *Participante Importante* é um tipo (relacionamento ISA) de participante, sendo que as funcionalidades associadas com esse ator são, na sua maioria, cobertas pelo ator *Participante de Reunião*. Conforme diretriz 4, consideramos que este ator é uma **especialização** de *Participante Importante* e definimos este relacionamento via mecanismo <<*generalization*>>, o qual é uma notação padrão em Diagramas de Casos de Uso em UML (Booch et al. 1999).

O próximo passo (2º passo – seção 5.2.2) na nossa proposta, visa descobrir e relacionar casos de uso aos atores descobertos.

6.2.1.2 Descobrindo casos de uso para os atores do sistema Agendador de Reunião

Concluída a descoberta dos atores para o sistema *Agendador de Reunião*, devemos encontrar casos de uso para estes atores com base no Modelo de Dependências Estratégicas da figura 13. Assim, inicialmente seguindo a diretriz 5 e observando a figura 13, podemos gerar a tabela 6. Esta tabela resume as informações necessárias para realizar o mapeamento de dependências em i* para casos de uso de atores.

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 6. Informação extraída dos Modelos de Dependências Estratégicas para derivar casos de uso para o sistema Agendador de Reunião

- Desta forma, para o ator ***Participante de Reunião***, observando-o como *Dependee* (diretriz 5, agendador de reunião → *dependum* → participante de reunião), podemos verificar a existência de duas dependências entre o ator e o sistema *Agendador de Reunião*: a tarefa *Entrar Datas Disponíveis* e o recurso *Concordância* (ver tabela 6). Estas dependências devem ser analisadas em um possível mapeamento para casos de uso. Inicialmente, a dependência do tipo tarefa *Entrar Datas Disponíveis* pode ser analisada sob a ótica da diretriz 5.2. Assim, seguindo essa diretriz, podemos considerar a necessidade de vários passos de interação entre participantes e o sistema *Agendador de Reunião* para entrar datas de agendamento disponíveis, entre os quais podemos citar: o fornecimento uma lista de datas preferidas e excluídas em um determinado formato por participantes, a validação destas datas pelo sistema, etc.. Assim, a tarefa *Entrar Datas Disponíveis* pode derivar o caso de uso ***Entrar Datas Disponíveis*** para o ator Participante de Reunião. Em seguida, a dependência do tipo recurso *Concordância* deve ser analisada. Seguindo a diretriz 5.3, podemos concluir que o principal objetivo da obtenção do recurso *Concordância* é obter

de cada participante a concordância em relação a uma data de agendamento proposta. Podemos considerar que neste processo de concordância, cada participante concordaria com a data proposta com certas restrições de horário ou duração da reunião. Ainda, a concordância poderia envolver uma análise de outras datas possíveis. Em outras palavras, o processo de obter concordância dos participantes em relação a datas de agendamento requer vários passos de interação entre os participantes e o sistema *Agendador de Reunião*. Assim o recurso Concordância pode ser mapeado para o caso de uso denominado de ***Concordância***, o qual incluirá os passos necessários para obter concordância de datas de agendamentos de participantes.

Observando agora o ator *Participante de Reunião* como *Depender* e o sistema computacional *Agendador de Reunião* como *dependee* (diretriz 6, participante de reunião $\rightarrow$ *dependum* $\rightarrow$ agendador de reunião) podemos encontrar a dependência do tipo recurso *DataProposta*. Utilizando agora a diretriz 5.3, verificamos que o maior objetivo com a obtenção deste recurso é que o sistema *Agendador de Reunião* deve fornecer uma data de agendamento para ser analisada pelos participantes. Esta data poderia ser fornecida, por exemplo, iniciando-se com participantes solicitando uma data proposta de agendamento, o sistema então procederia a escolha de uma data dentro de uma lista datas possíveis e consensuais sendo que esta escolha poderia envolver uma consulta prévia aos participantes, poderia também ser necessário realizar ajustes na data em ralação ao horário e tempo de duração da reunião e finalmente então, uma data consensual seria proposta, visando a concordância oficial por parte dos participantes. Assim, estes passos poderiam ser incluídos em um caso de uso

denominado de ***Data Proposta,*** derivado da necessidade da disponibilização do recurso *Data Proposta*.

- Para o ator ***Iniciador de Reunião***, podemos indicar alguns candidatos a casos de uso derivados dos relacionamentos de dependência entre este ator e o sistema *Agendador de Reunião* (ver tabela 6). Inicialmente, observando o ator *Iniciador de Reunião* como ***dependee*** (diretriz 5, agendador de reunião → *dependum* → iniciador de reunião), encontramos a dependência do tipo tarefa *Entrar Intervalo Datas*. Esta dependência pode ser analisada sob a ótica da diretriz 5.2. Seguindo esta diretriz, verificamos que fornecer um intervalo de datas para agendamento de reuniões pode incluir vários passos (sub-tarefas) tais como: associar intervalo de datas com reuniões específicas, estabelecer prioridades para certas reuniões, etc. Assim, a partir da tarefa *Entrar Datas Disponíveis,* podemos gerar o caso de uso ***Entrar Datas Disponíveis,*** o qual incluirá os passos necessários de interação entre iniciador de reunião e agendador de reunião para realizar essa tarefa.

  Por outro lado, analisando o ator *Iniciador de Reunião* como *depender* e o ator *Agendador de Reunião* como *dependee* (diretriz 6, iniciador de reunião → *dependum* → agendador de reunião), podemos observar a dependência do tipo tarefa *Reunião Agendada* (ver tabela 6). Esta dependência, como é do tipo objetivo, deve ser analisada sob a ótica da diretriz 5.1. Dessa análise, resulta a criação do caso de uso denominado ***Reunião Agendada*** para o ator Iniciador de Reunião. A idéia deste caso de uso é reforçada, se considerarmos também que vários passos são necessários para obter este objetivo, incluindo atividades tais como: obter datas disponíveis dos participantes, encontrar possíveis datas via

cruzamento de informações, propor uma data e obter concordância em relação a uma data proposta. Estas atividades podem ser observadas no Modelo de Razões Estratégicas da figura 14 e mais especificamente, na decomposição da tarefa *AgendarReunião Automaticamente,* a qual é a tarefa encarregada de satisfazer o objetivo *Reunião Agendada* e conseqüentemente, o caso de uso de mesmo nome originado desta dependência (ver diretriz 6).

É importante observar que a descrição do cenário primário (fluxo principal), de cada um dos casos de uso descobertos para os atores do sistema, pode dar origem a outros casos de uso. Isto exige uma análise posterior à descrição dos casos de uso, o que será realizado seguindo-se a diretriz 9. Neste momento, o que temos, são os casos de uso essenciais para o sistema Agendador de Reunião.

- Depois de derivados os casos de uso para o sistema Agendador de Reunião, podemos então seguindo a diretriz 7, classificar cada caso de uso e seu objetivo associado, como mostrado na tabela 7.

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

Tabela 7. Classificação de Objetivos para o sistema Agendador de Reunião.

Assim, fazendo uso das diretrizes propostas na seção 5.2.2, pudemos derivar casos de uso para o sistema Medi@ observando as dependências no Modelo de Dependências Estratégicas da figura 13. Os casos de uso descobertos incluem: ***Entrar Datas Disponíveis, Concordância*** e ***Data Proposta*** para o ator ***Participante de Reunião***

e os casos de uso ***Entrar Intervalo Datas*** e ***Reunião Agendada*** para o ator ***Iniciador de Reunião***. Continuando o processo de derivação, podemos agora iniciar a descrição/especificação dos cenários primário e secundários (fluxos principal e alternativos) dos casos de uso descobertos, com base nas informações disponíveis no modelo figura 14. Na próxima seção realizaremos este passo.

6.2.1.3 Especificando os casos de uso para o sistema Agendador de Reunião

Conforme diretriz 8, cabe neste passo observar o Modelo de Razões Estratégicas apresentado na figura 14, buscando informações que possam nos auxiliar na descrição dos casos de uso para o sistema Agendador de Reunião.

Por exemplo, vejamos a possibilidade de descrever o caso de uso ***Reunião Agendada*** para o ator ***Iniciador de Reunião*** (ver tabela 7), a partir da observação do modelo de Razões Estratégicas da figura 14. Este caso de uso representa o uso do sistema Agendador de Reunião pelo Iniciador de Reunião para agendar reuniões. Observando a figura 14 mais detalhadamente, verificamos que o sistema Agendador de Reunião satisfaz a dependência *Reunião Agendada* (*já mapeada para o caso de uso* ***Reunião Agendada***), através da tarefa *Agendar Reunião Automaticamente* e seus sub-componentes *Obter Datas Disponíveis, Encontrar Melhor Data, Propor Data e Obter Concordância.* Estes sub-componentes (conforme descrito na diretriz 8.1), podem ser mapeados para o cenário primário (fluxo principal) do caso de uso ***Reunião Agendada.*** Na figura 27, apresentamos uma possível especificação do caso de uso ***Reunião Agendada***, utilizando o *template* proposto por Cockburn (2000).

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 27. Especificação do caso de uso Reunião Agendada.

Assim, seguindo a diretriz 8, procedemos a observação do Modelo de Razões Estratégicas da figura 14, buscando quais elementos estão envolvidos na obtenção do objetivo *Reunião Agendada* pelo ator Agendador de Reunião. Este ator computacional tem a responsabilidade de levar a cabo este objetivo (através da interação com os atores do sistema), que foi anteriormente (ver tabela 7) mapeado para caso de uso. Assim,

observamos que os passos 1, 2, 3 e 4 são extraídos da ligação de decomposição da tarefa *Agendar Reunião Automaticamente,* cuja função é satisfazer a dependência do tipo objetivo *Reunião Agendada* (ver figura 14). O passo 1, neste caso de uso, é originado da sub-tarefa *Obter Datas Disponíveis*, estabelecendo que participantes da reunião, inicialmente, precisam fornecer as datas disponíveis para agendamento com base no intervalo de datas (precondição do caso de uso), previamente fornecido pelo iniciador de reunião. Notemos também que, a sub-tarefa *Obter Datas Disponíveis* tem uma dependência do tipo tarefa denominada *Entrar Datas Disponíveis* em relação ao ator participante de reunião, a qual tem sido previamente mapeada (ver tabela 7) para o caso de uso *Entrar Datas Disponíveis.* Com base nesta observação, o caso de uso ***Entrar Datas Disponíveis*** é incluído <<*include*>> no passo 1, pois representa os passos necessários (ações) para que os participantes forneçam as datas disponíveis para o agendamento de uma reunião.

O passo 2 foi derivado da observação do sub-objetivo *Encontrar Melhor Data*, estabelecendo que o sistema necessita encontrar a melhor data disponível para o agendamento. Observando a figura 14, verificamos que este objetivo pode ser alcançado (via ligação meio-fim), através da sub-tarefa *Cruzar Datas Disponíveis* (escolhida como a ação normal no cenário primário e assim descrito no passo 4). O passo 3 é derivado da sub-tarefa *Propor Data*, estabelecendo a necessidade do sistema propor uma data de agendamento a ser avaliada pelos participantes. Note que a sub-tarefa *Propor Data* satisfaz a dependência do tipo recurso *Data Proposta* mapeada anteriormente (ver tabela 7) para o caso de uso ***Data Proposta***. Com base nesta observação, o caso de uso ***Data Proposta*** é incluído <<*include*>> no passo 3 pois representa os passos necessários (ações) de interação entre o sistema agendador de reunião e participantes para propor uma data de agendamento.

Finalmente, o passo 4 é oriundo da observação da sub-tarefa *Obter Concordância*, estabelecendo a necessidade de se obter de cada participante a concordância em relação a data de agendamento proposta. Notemos também que, a sub-tarefa *Obter Concordância* possui uma dependência do tipo recurso denominada *Concordância* em relação ao ator participante de reunião, a qual tem sido previamente mapeada (ver tabela 7) para o caso de uso ***Concordância***. Com base nesta observação, o caso de uso ***Concordância*** é incluído <<*include*>> no passo 1, pois representa os passos necessários (ações) para que os participantes possam concordar em relação a uma data de agendamento proposta.

Assim, com as informações existentes no Modelo de Razões Estratégicas da figura 14, foi possível gerar a especificação para este caso de uso. Salientamos novamente que, quanto mais razões estratégicas forem modeladas nos Modelos SR, maior será a possibilidade de se aprofundar na descrição dos casos de uso descobertos.

Contudo, seguindo a diretriz 9, existe também a possibilidade de a partir das especificações de casos de uso já realizadas, evoluir na descoberta de novos casos de uso a partir da análise dos passos destas especificações. A seguir, realizamos a análise do caso de uso ***Reunião Agendada***.

- Inicialmente, verificamos que os passos 1, 3 e 4 do cenário primário (fluxo principal) do caso de uso ***Reunião Agendada*** já representam passos que incluem casos de uso. Assim, não cabe neste passo aplicar a diretriz 9.
- Já o passo 2 necessita ser investigado. Este passo/sentença é assim expresso no caso de uso (ver figura 27): "O sistema deve encontrar a melhor data para o agendamento cruzando as informações das datas disponíveis enviadas pelos participantes". O objetivo desta sentença,

respondendo à pergunta da diretriz 9.1, é "Encontrar Melhor Data". Observando a diretriz 9.2, deve-se investigar quais são os passos necessários (sub-objetivos) para se obter esse objetivo. Tipicamente, esta atividade é uma ação simples de Classificação/Cruzamento de Informações ("*Merge*") interna ao sistema Agendador de Reunião. Assim, não se justifica e não cabe aqui criar um caso de uso para este passo.

Desta forma, após termos descobertos os atores, bem como alguns casos de uso e suas especificações para o sistema Agendador de Reunião, podemos, conforme diretriz 10, gerar uma versão do Diagrama de Casos de Uso em UML para Agendador de Reunião. A figura 28 apresenta este diagrama. Salientamos que este diagrama, bem como as especificações de casos de uso realizadas, podem ser modificadas ou complementadas a medida que novos relacionamentos são encontrados. A nossa proposta visa encontrar os casos de uso essenciais do sistema e a partir deste ponto poder evoluir em relação à definição funcional completa de um sistema computacional.

Assim, observando-se os modelos organizacionais e adotando-se as diretrizes propostas neste trabalho, podemos extrair informações bastante úteis no desenvolvimento de Casos de Uso. Como dito anteriormente, a descoberta dos Casos de Uso, bem como a descrição dos mesmos pode ser melhorada a partir de uma observação mais detalhada dos modelos organizacionais. Na próxima seção, apresentamos as considerações finais deste capítulo.

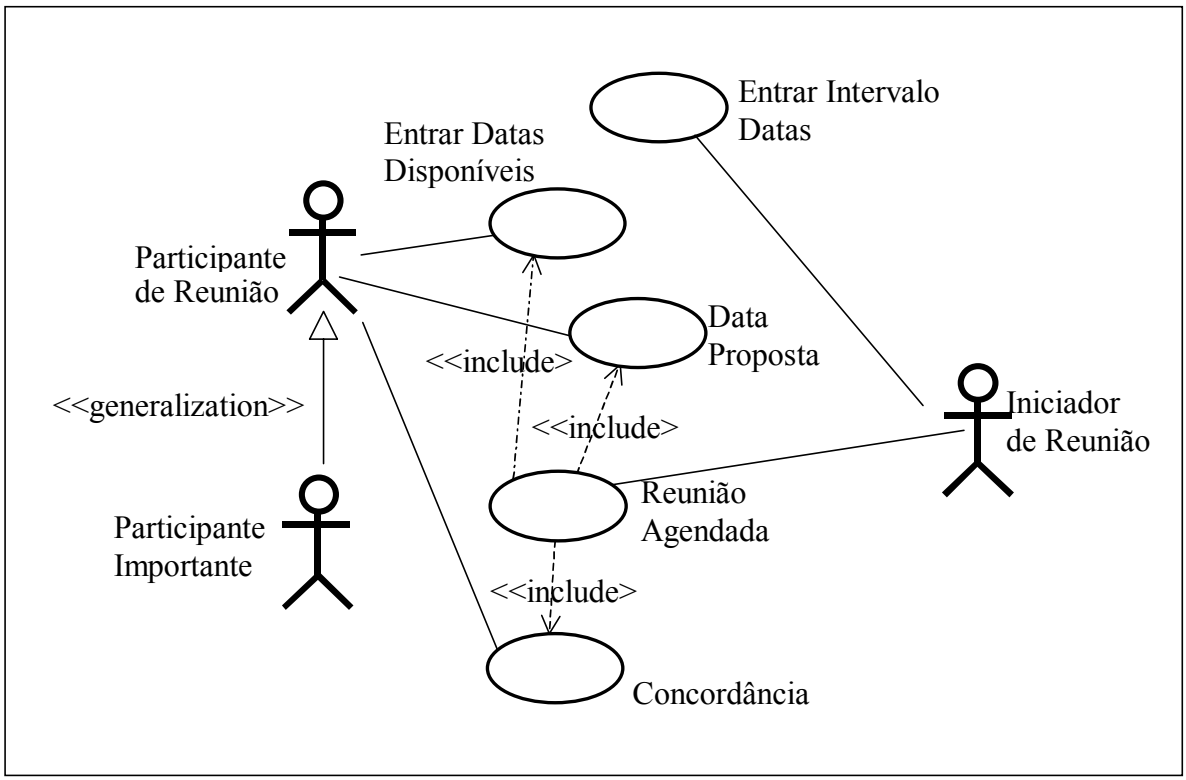

Figura 28. Diagrama de Casos de Uso para o sistema de Agendamento de Reuniões

6.3 Considerações Finais

Apresentamos neste capítulo, a nossa proposta de integração do *framework* i* com Casos de Uso. Definimos algumas diretrizes a serem aplicadas na descoberta de casos de uso em UML para o sistema pretendido, tomando como base os modelos organizacionais descritos em i*. As diretrizes foram aplicadas a dois estudos de caso: sistema de comércio eletrônico e sistema de agendamento de reuniões. Como resultado, obteve-se diagramas de casos de uso dos sistemas, bem como algumas descrições de casos de uso possíveis de serem extraídas dos modelos de razões estratégicas. Estes estudos de caso auxiliaram-nos na observação de que a integração de ambas as técnicas abordadas é viável e pode levar à elaboração de sistemas computacionais mais confiáveis e que atendam às reais necessidades da organização e/ou de usuários. Foi possível observar que o desenvolvimento de casos de uso pode ser mais efetivo, se

consideramos, de que forma o sistema em questão pode satisfazer os objetivos organizacionais. Verificamos também, que uma série de informações em i* podem ser mapeadas, sob o ponto de vista de uma análise orientada a objetivos, para casos de uso em UML.

No entanto, não incluímos na nossa proposta diretrizes mais sistemáticas para tratar requisitos não-funcionais, que podem ser derivados das dependências do tipo objetivo-soft em i*. Identificamos também que, o sucesso do mapeamento de i* para casos de uso depende de um bom conhecimento de engenheiros de requisitos de ambas as técnicas e principalmente, de estarem habituados com a representação gráfica empregada em i*. Além disso, temos observado que aplicando as diretrizes propostas neste trabalho, obtemos tipicamente os atores e casos de uso essenciais do sistema computacional pretendido. Torna-se necessário que, com base em uma análise da descrição dos casos de uso, bem como dos requisitos de sistemas computacionais expressos em outras formas tais como: questionários, entrevistas, etc, evoluamos em direção à definição de todos os casos de uso necessários para o sistema computacional pretendido. De forma geral, verificamos que como i* concentra-se nos relacionamentos "estratégicos" em um ambiente organizacional, é clara a necessidade de refinar e explorar, principalmente com a participação de usuários e clientes, os casos de uso e suas descrições associadas derivados dos modelos organizacionais, visando atingir descrições de casos de uso em um nível de abstração adequado (Cockburn 2000).

Outro aspecto importante é a limitação da técnica i* em relação à impossibilidade de expressar ordem no mecanismo de decomposição de tarefas (seção 4.1.2). Esta limitação impede que no modelo de razões estratégicas possamos indicar uma ordem às sub-tarefas que decompõem uma tarefa. Lembremos que no processo de derivação de descrições de casos de uso, estas decomposições (sub-tarefas) originam

passos nas descrições dos casos de uso. Como não se permite impor ordem às sub-tarefas em i*, cabe aos engenheiros de requisitos em uma análise adicional do domínio da aplicação, definir a seqüência, na qual as sub-tarefas serão mapeadas para passos em caso de uso.

No próximo capítulo apresentamos as considerações finais desta tese, bem como os trabalhos relacionados e futuros.

Capítulo 7

Considerações Finais e Trabalhos Futuros

Neste capítulo apresentamos na seção 7.1 as conclusões desta tese, descrevemos as contribuições realizadas e trabalhos relacionados nas seções 7.2 e 7.3, respectivamente, bem como sugerimos trabalhos futuros na seção 7.4.

7.1 Conclusões

Apresentamos nesta tese a proposta para integrar modelos organizacionais desenvolvidos com a técnica i* com cenários sob a forma de Casos de Uso. As diretrizes propostas foram aplicadas a dois estudos de caso, o primeiro relacionado a um sistema de comércio eletrônico e o segundo ao problema de agendamento de reuniões. Estes estudos de caso possibilitaram-nos mostrar que as informações existentes tanto no Modelo de Dependências Estratégicas quanto no Modelo de Razões Estratégicas podem servir de base para o desenvolvimento de Casos de Uso.

Além disso, permite-se que o engenheiro de requisitos, a partir de uma observação mais detalhada dos modelos organizacionais, possa optar pela melhor alternativa para desenvolver o sistema computacional, bem como concentrar-se nos Casos de Uso, que de fato, representam a satisfação dos objetivos de usuários e clientes. No desenvolvimento tradicional, casos de uso e cenários de uma forma geral, não consideram de forma eficiente, motivações, intenções e alternativas para o desenvolvimento de sistemas. O uso de modelos organizacionais auxilia neste sentido.

Com este tipo de abordagem, integrando modelos organizacionais e Casos de Uso, perguntas tais como: de que forma o sistema pretendido irá satisfazer os objetivos da organização, por que ele é necessário, quais as alternativas existentes e quais as implicações das alternativas para as várias partes interessadas, podem ser melhor

respondidas, tratadas e as soluções incorporadas no desenvolvimento de sistemas computacionais.

7.2 Contribuições

De forma mais específica, descrevemos abaixo as principais contribuições deste trabalho:

- entendimento das razões ou "porquês": vários pesquisadores (Anton 1997) (Castro et al. 2001) (Castro et al. 2001a) (Chung et al. 2000) (Dardene et al. 1993) (Rolland et al. 1998) (Yu 1995) (Yu et al. 1998) têm considerado objetivos em várias diferentes áreas da Engenharia de Requisitos. Abordagens orientadas a objetivos utilizadas para elicitação e descrição de requisitos podem ser contrastadas com técnicas que tratam requisitos como consistindo apenas de processos e dados, como ocorre na tradicional análise estruturada de sistemas ou "objetos", como ocorre na conhecida análise orientada a objetos. Tipicamente, estas técnicas tradicionais não capturam explicitamente os "porquês" e "comos" em termos de objetivos. Entender principalmente os "porquês", que expressam as razões e justificativas para sistemas computacionais propostos é considerado, atualmente, um aspecto essencial para o sucesso na elaboração de documentos de requisitos para estes sistemas computacionais.

- melhoria na avaliação da dinâmica dos processos de negócio: os relacionamentos entre sistemas e seus ambientes podem também ser expressos em termos de relacionamentos baseados em objetivos. Isto é parcialmente motivado pelos atuais ambientes organizacionais e de negócios cada vez mais dinâmicos, nos quais sistemas computacionais são

crescentemente utilizados para, fundamentalmente, modificar processos de negócios (Yu et al. 1998). Derivar casos de uso a partir dos relacionamentos em i*, permite que possamos rastrear e avaliar os impactos que as mudanças nas organizações (principalmente mudanças nos objetivos em relação a sistemas pretendidos) causam nos requisitos funcionais e não-funcionais dos sistemas computacionais. Tipicamente, uma mudança nos processos de negócio em um ambiente organizacional modifica ou cria novos relacionamentos do tipo tarefa, recurso, objetivo ou objetivo-soft entre atores nos modelos organizacionais em i*. Como o nosso processo de derivação de casos de uso toma como base estes elementos, podemos então analisar se determinados relacionamentos (tarefa, objetivo, recurso, objetivo-soft) , previamente mapeados para casos de uso, foram modificados em virtude de alterações nos processos organizacionais e como estas mudanças afetam os casos de uso derivados para sistemas computacionais pretendidos.

- melhoria no processo de desenvolvimento de casos de uso: alguns dos principais problemas e deficiências associados com o processo de desenvolvimento de casos de uso (Lilly 1999), podem ser parcialmente solucionados utilizando a nossa proposta. Uma descrição mais detalhada dos problemas com casos de uso apresentados em Lilly (1999), bem como possíveis soluções adotando-se a nossa proposta, foram descritos no capítulo 5.

- melhoria na descrição dos requisitos: elicitar e especificar requisitos de sistemas computacionais, observando os objetivos de atores em relação ao sistema a ser desenvolvido, é uma forma de tornar requisitos mais claros (Yu

et al. 1998). A partir de i* podemos derivar estes objetivos, associá-los com atores do sistema e então operacionalizá-los e refiná-los em requisitos descritos em casos de uso.

Contudo, com base nos estudos de caso realizados temos verificado algumas dificuldades e deficiências da proposta. Uma dificuldade decorre do fato de que não são apresentadas na literatura, heurísticas que auxiliem engenheiros de software no desenvolvimento dos modelos organizacionais usando i*. A técnica i* tem um rico poder de expressão e é de fácil entendimento, mas exige um tempo de dedicação para familiarizar-se completamente com os conceitos que a mesma suporta. Contudo, temos observado que após o correto entendimento da técnica i*, o processo de mapeamento torna-se bastante simples, útil, vantajoso e considerado possível de ser realizado em um tempo aceitável por desenvolvedores. Outro aspecto importante na nossa proposta é a dependência direta existente em relação à qualidade e completude dos Modelos de Dependências Estratégicas (SR) e Modelos de Razões Estratégicas (SR) em i*. O mapeamento de elementos em i* para casos de uso consistentes depende diretamente da qualidade, consistência e nível de detalhes associados com as informações descritas nos modelos organizacionais em i*.

Temos também observado que a técnica i* permite expressar várias informações, tais como: razões internas (incluindo tarefas, recursos, objetivos, objetivos-soft e ligações do tipo meio-fim e de decomposição de tarefa) associadas a atores organizacionais que não representam o sistema computacional. Algumas destas informações não estão associadas diretamente a funcionalidades de sistemas computacionais pretendidos e, portanto, não são incluídas no nosso processo de mapeamento de i* para casos de uso. No entanto, entendemos que estas informações são

essenciais para o entendimento do ambiente organizacional, no qual o software irá operar, bem como para avaliar as mudanças que ocorrerão nos processos organizacionais em virtude do novo software a ser desenvolvido.

Outro ponto importante observado é a necessidade de criação de uma linguagem que permita definir restrições na técnica i*, como por exemplo, impor ordenamento (seqüência) na decomposição de uma tarefa. Outras técnicas, tais como: fluxos de trabalho (*workflow*) e diagramas de atividade em UML, permitem estabelecer ordem na realização de tarefas, o que facilita o entendimento de como tarefas podem ser relaizadas. Consideramos este aspecto importante, principalmente porque no processo de derivação de descrições de casos de uso a partir da decomposição de tarefas representadas no modelo de Razões Estratégicas , exige-se um trabalho extra de engenheiros de requisitos em relação à definição da ordem, na qual sub-tarefas são mapeadas para passos em casos de uso (veja diretriz 8.1).

7.3 Trabalhos Relacionados

Alguns dos principais trabalhos relacionados incluem propostas centradas no desenvolvimento de software orientado a requisitos/objetivos, apresentadas no *framework Tropos* (Mylopoulos et al. 2000) (Castro et al. 2001) (Castro et al. 2002) e a integração de i* com diagramas precisos de UML (Alencar et al. 2000). Estes trabalhos defendem a idéia de que modelos organizacionais são fundamentais para poder evoluir em direção ao desenvolvimento de softwares mais confiáveis e que satisfaçam as reais necessidades de usuários e organizações. Outros trabalhos relacionados também incluem metodologias que combinam cenários e objetivos. Algumas destas metodologias foram resumidamente descritas no capítulo 3. Outros trabalhos com este enfoque incluem o método *Scenic* (Potts 1997), o qual utiliza o refinamento de objetivos

e análise de cenários como principais estratégias metodológicas. O método *Scenic* inclui estratégias sistemáticas para identificar atores, objetivos, tarefas e obstáculos na descrição de requisitos de sistemas. Nesta mesma linha de pesquisa, temos também o trabalho de Anton et al. (2001), no qual o método *GBRAM* (Anton 1997) é usado para derivar objetivos a partir de uma especificação de requisitos baseada em casos de uso. Os desafios e riscos associados com o desenvolvimento de sistemas de qualidade durante a análise de cenários e objetivos também são descritos nesse trabalho. Contudo, estas abordagens não consideram modelos organizacionais para derivar objetivos e cenários de sistemas computacionais pretendidos.

7.4 Trabalhos Futuros

Como trabalhos futuros, temos estabelecido a prioridade de descrever diretrizes mais sistemáticas para auxiliar engenheiros de requisitos a explorar, de forma mais efetiva, os requisitos não-funcionais definidos nos modelos em i* (principalmente à luz da proposta de Chung et al. (2000)) e então complementar a descrição de requisitos funcionais descritos em casos de uso. Outro aspecto detectado como trabalho futuro é fundamentado na viabilidade observada de poder aplicar a nossa proposta ao método L'Ecritorire (Rolland et al. 1998), pois há a clara possibilidade de explorar a técnica i* como fonte de informação para a descoberta de objetivos e cenários, conforme propõe o método. Da mesma forma, poderíamos utilizar os elementos em i* para enriquecer o modelo de cenários proposto por Leite et al. (1997). Além destes aspectos, pretendemos também realizar mais estudos de caso, bem como prover um suporte computacional para a nossa proposta.

Referências Bibliográficas

Alencar, F., Castro, J., Cysneiros, G., Mylopoulos, J., "From Early Requirements Modeled by i* Technique to Later Requirements Modeled in Precise UML", In Proceedings of the "III Workshop de Engenharia de Requisitos", pp 92-108, Rio de Janeiro, Julio, (2000).

Alencar, F., "A Modelagem Organizacional e a Especificação Formal", Centro de Informática, Universidade Federal de Pernambuco, Tese de Doutorado, Dezembro, (1999).

Alencar, Fernanda M., Souza, Fabrizia M.,Castro, Jaelson B., "Modelagem Organizacional: Análise Comparativa das Técnicas I* e Bubenko". In Proceedings of IDEAS'99 - II Ibero American Workshop on Requirements Engineering and Software Environment. ISBN 9968-32-000-5. San José, Costa Rica – March, (1999), pp. 326-- 337.

Anton, A., *Goal identification and refinement in the specification of software-based information systems*. Phd Thesis, Georgia Institute of Technology, Atlanta, GA, June, (1997).

Anton, A., Carter, R.A., Dagnino, A., Dempster, J.H., Siege, D.F., "Deriving Goals from a Use Case Based Requirements Specification", Requirements Engineering Journal, Springer-Verlag, Volume 6, May (2001), pp. 63-73.

Booch, G., Jacobson, I., Rumbaugh, J., *The Unified Modeling Language User Guide*, Addison-Wesley, (1999).

Booch,G., *The Booch Method: Process and Pragmatics*, Santa Clara, Calif.: Rational, (1992).

Boehm, B. W, *Software Risk Management*, IEEE Computer Society Press, Washington, 1989.

Brooks, F. P., "No silver bullet; essence and accidents of software engineering", Computer, Vol. 20, No. 4, IEEE, pp. 10-19, April, (1987).

Breitman, K.K., Leite, J.C.S.P., "A Framework for Scenario Evolution", Third International Conference on Requirements Engineering – III, IEEE Computer Society Press. Los Alamitos, CA, U.S.A, (1998), pp 214 – 221.

Bubenko, J. A., "Extending The Scope of Information Moldeling", Proc. 4th Int. Workshop on the Deductive Approach to Information Systems and DataBases, Lloret-Costa Brava, Catalonia, Sept. 20-22, (1993), pp 73-98.

Bubenko , J, A., Kirikowa, M., "Worlds" in Requirements Acquisition and Modelling, 4th European - Japanese Seminar on Information Modelling and Knowledge Bases, Kista, Sweden, edited by H. Kangassalo and B. Wangler. (1994).

Bubenko, J., *Challenges in Requirements Engineering*, Second IEEE International Symposium on Requirements Engineering, IEEE Computer Society Press. York, England, 1995.

Bortoli. L.A.D., Price, A.M.A., O Uso de Workflow para Apoiar a Elicitação de Requisitos, In: III Workshop de Engenharia de Requisitos, WER'00, 13 e 14 de Julho, Rio de Janeiro, pp 22-37.(2000).

Castro, J., Kolp, M. and Mylopoulos, J., "Developing Agent-Oriented Information Systems for the Enterprise", Proceedings of the Second International Conference on Enterprise Information Systems (ICEIS00), Stafford, UK, July, (2000).

Castro, Jaelson F., Kolp, M., Mylopoulos, J., "A Requirements-Driven Development Methodology", In: CAISE'01, Proceedings of the 13th Conference on Advanced

Information Systems Engineering. Heildelberg, Germany: Springer Lecture Notes in Computer Science LNCS 2068, (2001), pp. 108-123.

Castro, J., Alencar, F., Cysneiros, G., Mylopoulos, J., "Integrating Organizational Requirements and Object Oriented Modeling", In Proceedings of the Fifth IEEE International Symposium on Requirements Engineering - RE'01, August 27-31, Toronto, (2001a), pp. 146-153.

Castro, J., Kolp, M., Mylopoulos, J., "Towards Requirements-Driven Information Systems Engineering: The *Tropos* Project", 35 pages. In *Information Systems*, Elsevier, Amsterdam, The Netherlands, (2002).

Caliber-RM, www.tbi.com (acessado em 30/11/2002)

Chung, L., Nixon, B.A.,Yu, E., Mylopoulos, J., *Non-Functional Requirements in Software Engineering (Monograph)*, Kluwer Academic Publishers, 472 pp, (2000).

Clarke, Edmund M., Jeanette M. Wing et al., "Formal Methods: State of the art and future directions". ACM Computing Surveys n. 4, vol. 28, (1996).

Coleman D., P. Arnold, S. Bodoff, C. Dollin, H. Gilchrist, F. Hayes, P. Jeremaes, *Object-oriented development: The Fusion Method*, Prentice-Hall, (1994).

Cockburn, A., *Writing Effective Use Cases*, Humans and Technology, Addison-Wesley, (2000).

Dardene,A., Lamsweerde, V., Fikas, S., "Goal-Directed Requirements Acquisition", Science of Computer Programming, 20, pp. 3-50, 1993.

D'Souza , Desmond F., A.C. Wills, *Objects, Components and Frameworks with UML: The Catalysis Approach,* Addison-Wesley, November, (1998).

Davis, A., *Software Requirements: Objects, Functions, and States*, PTR Prentice Hall, Englewood Cliffs, New Jersey, USA, (1993).

Diller, A., *Z: An Introduction to Formal Methods*, John Wiley & Son Ltd, (1994)

Dubois, E., P. Heymans, "Scenario Based Techniques for supporting the elaboration and the validation of Formal Requirements", RE Journal, (1998).

DeMarco, T., Structured Analysis and System Specification, Englewood Cliffs, New Jersey: Prentice-Hall, (1979).

Dobson, J. E, Blyth, A.J. C., Chudge, J. and Strems, R., "The ORDIT approach to organisational requirements", In M. Jirotka and J. Goguen (Eds.), Requirements Engineering, Academic Press Ltd, London, (1994).

Eriksson, Hans-Erik., Penker, Magnus., *Business Modeling with UML: business patterns a work*, John Wiley & Sons, (2000).

Estrada, H, Martínez, A., Pastor, O., Ortiz, J., Rios, O.A., "Generación Automática de un Esquema Conceptual OO a Partir de un Modelo Conceptual de Flujo de Trabajo", In: IV Workshop de Engenharia de Requisitos, WER'01, 22 e 23 de Novembro, Buenos Aires, (2001).

Estrada, H, Martínez, A., Pastor, O., Sanches, J., "Generación de Especificaciones de Requisitos de Software a partir de Modelos de Negócios: un Enfoque basado en Metas", In: V Workshop de Engenharia de Requisitos, WER'02, 11 e 12 de Novembro, Valencias, (2002).

Fantechi, A., Gnesi, S., Lami, G., Maccari, A., "Application of Linguistic Techniques for Use Case Analysis", In: IEEE Joint International Requirements Engineering Conference, RE'02, University of Essen, Germany, September, 9-13, (2002), pp 157-164.

Fiorini , Soeli., Leite, J.C.S.P., Macedo-Soares, T. D., "Integrando Processos de Negócio à Elicitação de Requisitos", Anais do IX Simpósio Brasileiro de Engenharia de Software, SBC, Out., pp. 379--394. (1995).

Fiorini, S.T., Processos de Negócio e Hipertexto: uma proposta para a elicitação de requisitos, Dissertação de Mestrado, Departamento de Informática, PUC-Rio, (1999).

Fiorini, S.T., Leite, J.C.S.P, Macedo-Soares, T.D.L., "Integrating Business Processes with Requirements Elicitation", *Proceedings of the 5th Workshops on Enabling Technologies:Infrastructure for Collaborative Enterprises* (WET ICE'96), 1996.

Goguen, J. and Linde, C., "Techniques for Requirements Elicitation", Proceedings of the IEEE International Symposium on Requirements Engineering. San Diego, CA, USA, IEEE Computer Society Press, Los Alamitos, (1993), pp. 152-164.

Hadad, G. D.S., Doorn, J.H., Kaplan, G.N., Leite, J.C.S.P., "Enfoque Middle-Out en la Construcción e Integración de Escenarios", nos anais do Workshop em Engenharia de Requisitos, Buenos Aires, Argentina, 9-10 September, (1999).

Hughes, J. et al, "The Role of Ethnography in Interactive Systems Design", Interactions, ACM Press, April, (1995), pp. 56-65.

Haumer, P., Pohl, K., Weidenhau´pt, K., "Requirements Elicitation and Validation with Real World Scenes", IEEE Transactions on Software Engineering, Vol 24, No 12, Special Issue on Scenario Management, December, (1998).

Humphrey, W.S., *Managing the Software Process*, Addison Wesley, (1989).

Hofman, H., *Requirements Engineering: A Survey of Methods and Tools*, Institut für Informatik der Universität Zürich, Nr. 93.05, 1993.

Jacobson, I., Booch, G., Rumbaugh, J., *The Unified Software Development Process*, Addison-Wesley (1999).

Jacobson, I., *Object Oriented Software Engineering: A Use Case Driven Approach,* Addison-Wesley (1995).

Jackson, M, *Software Requirements and Specifications: A Lexicon of Practice, Principles and Prejudices*, 1ed. Addison-Wesley, Massachusetts, USA, 1995.

Jones, C.B., *Systematic Software Development Using VDM*, Prentice-Hall, (1990)

Kaiva, H., Horai, H., Saeki, M., "AGORA: Attributed Goal Oriented Requirements Analysis Method", In: IEEE Joint International Requirements Engineering Conference, RE'02, University of Essen, Germany, September, 9-13, (2002), pp 13 - 22.

Kauppinen, M., Kujala, S., Aaltio, T., Lehtola, L., "Introducing Requirements Engineering: How to Make a Cultural Change Happen in Practice", In: IEEE Joint International Requirements Engineering Conference, RE'02, University of Essen, Germany, September, 9-13, (2002), pp. 43 - 51.

Kotonya, G., Somerville, I., *Requirements Engineering: Processes and Techniques*, John Wiley & Sons, (1997).

Kozlenkov, A., Zisman, A., "Are Their Design Specifications Consistent with Our Requirements?", In: IEEE Joint International Requirements Engineering Conference, RE'02, University of Essen, Germany, September, 9-13, (2002), pp 145-154.

Kruchten, Philippe, *The Rational Unified Process*, Addison-Wesley Pub Co, ISBN: 0201707101, 2nd edition, (2000)

Lamsweerde, A. van, "Requirements Engineering in the Year 00: A Research Perspective", 22nd Proceedings of International Conference on Software Engineering, Limerick, Ireland. June, (2000).

Leite, J.C.S.P, Rossi, G., Balaguer, F., Maiorana, V., "Enhancing a requirements baseline with scenarios". In Proceedings of the Third IEEE International Symposium on Requirements Engineering – RE'97, IEEE Computer Society Press, January, (1997), pp. 44-53.

Leite, J., "A Survey on Requirements Analysis", Technical Report, Department of Information and Computer Science, California, 1987.

Leite, J. C.S.P., Leonardi, M.C., "Business Rules as Organizational Policies", Proceedings of the International Workshop on Software Specification and Design, IEEE Computer Society Press, pp. 68-76 (1998).

Leonardi, M. C., Leite, J.C.S.P., Rossi, G., "Estratégias para la Identificación de Reglas de Negocio", Anais do XII Simpósio Brasileiro de Engenharia de Software, Sociedade Brasileira de Computação, Out., pp. 53—67.(1998)

Lilly, S., "Use Case Pitfalls: top 10 problems from Real projects using use cases", In: Proceedings, technology of object oriented languages and systems, 1-5 August (1999), pp. 174-183.

Loucopoulos, P. e Karakostas, V., *System Requirements Engineering*, McGraw-Hill, 1995.

Macaulay, L., *Requirements Engineering*, Springer, London, 1996

Mylopoulos, J., Chung, L., Yu, E., "From Object-Oriented to Goal-Oriented Requirements Analysis," *Communications of the ACM*, **42**(1), January 1999, pp. 31-37.

Mylopoulos, J., Castro, J., "Tropos: A Framework for Requirements-Driven Software Development" , Brinkkemper, J. and Solvberh, A. (eds), *Information Systems Engineering: State of Art and Research Themes*, Lectures Notes in Computer Science, Springer-Verlag, June (2000).

Mylopoulos, J., "Capturing Intentions in Requirements Engineering: A modeling Perspective", Invited Talk at the IFIP W2.9 meeting, Bramshill, UK, March, (1995).

Oskarsson, O., Glass, R. L., *An Iso 9000 Approach to Building Quality Software*, Prentice Hall, (1995)

Penadés, M.C., Canós, J. H., Sánchez, J., "Automatic Derivation of Workflow Specifications from Organizational Structures and Use Cases", In: IV Workshop de Engenharia de Requisitos, WER'01, 22 e 23 de Novembro, Buenos Aires, (2001).

Pohl, K., Haumer, H., "Modelling Contextual Information about Scenarios", Proc. Third International Wokshop Requirements Engineering: Foundations of Software Quality REFSQ'97, pp 187-204, Barcelona, June, (1997).

Potts, C., "Fitness for Use: the System Quality that Matters Most", Proc. Third Int'l Workshop Requirements Engineering: Foundations of Software Quality REFSQ'97, pp. 15-28, Barcelona, June, (1997).

Potts, C., "ScenIC: A Strategy for Inquiry-Driven Requirements Determination", In Proceedings of the Fourth IEEE International Symposium on Requirements Engineering - RE'99, June 7-11, Ireland, (1999).

QSSRequireit, http://www.qssrequireit.com (acessado em 30/11/2002)

Quality Systems and Software (1999). DOORS http://www.qss.co.uk (acessado 30/11/2002)

Ralyté, Jolita., Rolland, C., Plihon, V., "Method Enhancement With Scenario Based Techniques", In *Proceedings* of CAISE 99, 11th Conference on Advanced Information Systems Enguneering Heidelberg, Germany, June 14-18, (1999), (CREWS Report Series 99-10).

Ralyté, Jolita., "Reusing Scenario Based Approaches in Requirements Enginnering Methods: Crews Method Base", In *Proceedings* of REP'99, 1st International Workshop on the Requirements Engineering Process, Florence, Italy, September (1999), (CREWS Report Series 99-12).

Rational Corporation (1999) Requisite Pro http://www.rational.com (acessado em 30/11/2002).

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).

Rosca, D., Greenspan, S., Feblowitz, M., Wild, C., "A Descision Making methodology in Support of the Business Rules Lifecycle", Proceedings of IEEE International Symposium on Requirements Engineering - RE97, pp.236-246 Jan. 1997.

Rumbaugh, J., Blaha M., Premerlani, W., Eddy, F., Lorensen, W., *Modelagem e Projetos Baseados em Objetos*, editora Campus, (1994).

Santander, V. F., Castro, J.F., "Deriving Use Cases from Organizational Modeling", In: IEEE Joint International Requirements Engineering Conference, RE'02, University of Essen, Germany, September, 9-13, (2002), pp. 32 – 39.

Santander, V. F., Castro, J. F. B. "Integrating Use Cases and Organizational Modeling", In: Brazilian Symposium on Software Engineering, SBES'02, Gramado, Rio Grande do Sul, October, 16-18, (2002a).

Santander, V. F., Castro, J. F., "Developing Use Cases from Organizational Modeling", In: IV Workshop de Engenharia de Requisitos, WER'01, 23 e 24 Novembro, Buenos Aires, (2001).

Schneider, G., Winters, J. P., *Applying Use Cases: a practical guide*, Addison Wesley, (1998).

Sommerville, I. et al, "Integrating Ethnography into the Requirements Engineering Process", Proceedings of the 1st IEEE International Symposium on Requirements Engineering. San Diego, CA, USA, IEEE Computer Society Press, Los Alamitos, (1993), pp. 165-173;

Sommerville, I., Viller, S., "Social Analysis in the Requirements Engineering Process: from Ethnography to Method", Proceedings of the 4th IEEE International Symposium on Requirements Engineering. Limerick, Ireland, IEEE Computer Society Press, Los Alamitos, (1999), pp. 6-13;

Sommerville, Ian., Peter Sawyer, *Requirements Engineering: A good practice guide*, John Wiley & Sons, (1997), ISBN: 0 471 97444 7.

Sutcliffe, A., Gregoriades, A., "Validating Functional Requirements with Scenarios", In: IEEE Joint International Requirements Engineering Conference, RE'02, University of Essen, Germany, September, 9-13, (2002), pp. 181-188.

Sufcliffe, A. G., "Scenario-based Requirements Analysis", Requirements Engineering Journal, Vol (3) No 1, Springer Verlag, (1998).

Spivey, M., Spivey, J.M., *The Z Notation: A Reference Manual*, Prentice-Hall, (1992)

Tawni, Mustapha., Souveyet, C., "Guiding Requirement Engineering with a Process Map", Proceedings of MFPE'99: 2nd International Workshop on Many Facets of Process Engineering, Gammarth, Tunisia, 12-14 May (1999).

Toranzo, Marco A., "Uma Proposta para Melhorar o Rastreamento de Requisitos de Software", Centro de Informática, Universidade Federal de Pernambuco, Tese de Doutorado, Dezembro, (2002).

Woo, H., Robinson, W., "Reuse of Scenario Specifications Using an Autometed Relational learner: A Lightweight Approach", In: IEEE Joint International Requirements Engineering Conference, RE'02, University of Essen, Germany, September, 9-13, (2002), pp. 173-180.

Yu, Eric, *Modelling Strategic Relationships for Process Reengineering,* Phd Thesis, University of Toronto, (1995).

Yu, E., Bois, P., Mylopoulos, J., "*From Organization Models to System Requirements – A Cooperative Information Systems"*, CoopIS-95, Vienna (Austria), (1995).

Yu, E., Mylopoulos, J., "Why Goal-Oriented Requirements Engineering", Proc. Fourth International Workshop Requirements Engineering: Foundations of Software Quality REFSQ'98, Pisa, June, (1998), pp 15-22.

Yu, E., "*Towards Modelling and Reasoning Support for Early-Phase Requirements Engineering",* Proceedings of IEEE International Symposium on Requirements Engineering - RE97, Jan, (1997), pp.226-235.

ISO/IEC 9126 Standard for Information Technology, Software Product Evaluation - Quality Characteristics and Guidelines for their use, Geneve (1991).

NBR ISO/IEC 12207 "Tecnologia de Informação – Processos de Ciclo de Vida de Software", Genebra, (1995).

ISO/IEC TR 15504 (SPICE Project) http://www-sqi.cit.gu.edu.au/spice/ (acessado em 30/11/2002).

IEEE, *IEEE Software Engineering Standards Collection*, Computer Society Press, 1997

IEEE Std. 830, *IEEE Guide to Software Requirements Specification*, The Institute of Electrical and Electronics Engineers, New York, EUA, (1984).

Zave, P., *Classification of Research Efforts in Requirements Engineering*, ACM Computer Surveys, vol 29, n ° 4, (1997).