Herbet de Souza Cunha

Uso de Estratégias Orientadas a Metas para Modelagem de Requisitos de Segurança

Dissertação de Mestrado

Dissertação apresentada como requisito parcial para obtenção do grau de Mestre pelo Programa de Pós-graduação em Informática do Departamento de Informática da PUC-Rio.

Orientador: Prof. Julio César Sampaio do Prado Leite

Rio de Janeiro

abril de 2007

Herbet de Souza Cunha

Uso de Estratégias Orientadas a Metas para Modelagem de Requisitos de Segurança

Dissertação apresentada como requisito parcial para obtenção do grau de Mestre pelo Programa de Pós-graduação em Informática do Departamento de Informática da PUC-Rio. Aprovada pela Comissão Examinadora abaixo assinada.

Prof. Julio Cesar Sampaio do Prado Leite
Orientador
Departamento de Informática – PUC-Rio

Prof. Arndt Von Staa
Departamento de Informática – PUC-Rio

Profa. Vera Maria Benjamim Werneck
Universidade do Estado do Rio de Janeiro - UERJ

Prof. José Eugênio Leal
Coordenador Setorial do Centro
Técnico Científico – PUC-Rio

Rio de Janeiro, 16 de abril de 2007

Herbet de Souza Cunha

Graduou-se em Bacharelado em Ciências da Computação pela Universidade Federal de Pernambuco em agosto de 2000. Área de interesse acadêmico: Engenharia de Software, mais especificamente a sub-área de Engenharia de Requisitos. Atualmente é funcionário da área de Tecnologia da Informação da Petróleo Brasileiro SA (PETROBRAS) atuando na área de arquitetura de informações e metadados.

Ficha Catalográfica

Cunha, Herbet de Souza

Uso de estratégias orientadas a metas para modelagem de requisitos de segurança / Herbet de Souza Cunha; orientador: Julio César Sampaio do Prado Leite. – 2007.
148 f. : il. ; 30 cm

Dissertação (Mestrado em Informática)– Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro, 2007.
Inclui bibliografia

1. Informática – Teses. 2. Requisitos. 3. Segurança. I. Leite, Julio César Sampaio do Prado. II. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Informática. III. Título.

CDD: 004

Para meus pais, Nalva e Milton, pelo
incentivo e confiança.

Agradecimentos

A Julio César Sampaio do Prado Leite pelas orientações e paciência.

A Antônio de Pádua Albuquerque Oliveira pela colaboração.

Aos amigos da Petrobras pelo incentivo e força.

A Petrobras pelo apoio.

Resumo

Cunha, Herbet de Souza; Leite, Julio Cesar Sampaio do Prado. **Uso de estratégias orientadas a metas para modelagem de requisitos de segurança**. Rio de Janeiro, 2007. 148p. Dissertação de Mestrado - Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro

Adicionar requisitos de segurança às arquiteturas de software após elas terem sido construídas é uma tarefa bastante difícil. Conceitos de segurança devem permear todo o ciclo de desenvolvimento do software, desde a engenharia de requisitos, passando por desenho (design), implementação, testes e distribuição. Este trabalho apresenta uma abordagem para modelagem de requisitos de segurança, especialmente os requisitos de confidencialidade e consistência das informações, baseada em estratégias orientadas a metas, trazendo a questão da segurança para o início do ciclo de desenvolvimento de software. São apresentados também os resultados da aplicação desta abordagem em um estudo de caso.

Palavras-chave

Requisitos, Segurança

Abstract

Cunha, Herbet de Souza; Leite, Julio Cesar Sampaio do Prado. **Uso de estratégias orientadas a metas para modelagem de requisitos de segurança**. Rio de Janeiro, 2007. 148p. Master degree thesis – Computer Science Department, Pontifícia Universidade Católica do Rio de Janeiro

Adding security requirements to software architectures after they are built is a hard work. Security concepts have to cross the whole software development cycle, from requirement engineering to deployment, passing by design, coding and test. This work presents an approach to security requirements modeling, mainly the information confidentiality and consistency, based on goal oriented strategies, bringing the security issues to the beginning of the software development cycle. It also presents the results of this approach in a case study.

Keywords

Requirements, Security.

Sumário

Lista de Figuras

Lista de Tabelas

1
Introdução

O objetivo deste capítulo é estabelecer o contexto da pesquisa realizada. Ao longo deste capítulo serão apresentadas: a motivação da pesquisa, o objetivo, trabalhos relacionados, conceitos utilizados e, por fim, um resumo da organização da dissertação.

1.1 Motivação

Existe um consenso tanto na comunidade acadêmica quanto nas empresas, que a segurança dos sistemas de software é importante. A sociedade moderna depende criticamente de uma variedade destes sistemas de software [21]. Com a popularização da internet, o número de incidentes de segurança reportados tem crescido rapidamente nos últimos anos [23]. A falha de um software pode provocar danos de abrangência variada, desde uma simples quebra do mecanismo de proteção de cópias de um 'jogo' até um desastre provocado por uma entrada não autorizada no sistema de controle de uma planta de usina nuclear. Em virtude disto, desenvolvedores de software devem pensar não apenas nos usuários, mas também nos adversários [21].

Os conceitos de segurança devem permear todo o ciclo de desenvolvimento do software, desde a engenharia de requisitos, passando por desenho (design), implementação, testes e distribuição. É bem entendido pelos desenvolvedores de software que adicionar requisitos de qualidade (não-funcionais), inclusive requisitos de segurança, às arquiteturas de software após estas serem feitas é difícil ou impossível [21]. É fundamental, portanto, que tais requisitos sejam endereçados o mais cedo possível.

Giorgini [28] destaca ainda a importância de se capturar requisitos de segurança de alto nível, e de tornar claros os motivos do uso de mecanismos de proteção como criptografia e autorização de acesso.

1.2 Objetivo

Nosso trabalho apresenta uma abordagem orientada a metas para definição (elicitação, modelagem e análise) de requisitos de segurança específicos de aplicação. Através desta abordagem, são construídos modelos intencionais para os requisitos do domínio e para os requisitos de segurança.

A elaboração destes modelos, juntamente com a documentação complementar proposta, tem por objetivo atacar os seguintes pontos:

- ⇒ Elicitar os requisitos de segurança de alto nível;
- ⇒ Inter-relacionar os requisitos de segurança com os demais requisitos;
- ⇒ Tornar explícita a motivação dos requisitos de segurança (por que eles foram incorporados aos demais requisitos);
- ⇒ Identificar atacantes potenciais;
- ⇒ Identificar possíveis intenções maliciosas dos atacantes;
- ⇒ Identificar ameaças (medidas de ataque);
- ⇒ Identificar pontos de vulnerabilidade do sistema;
- ⇒ Avaliar alternativas para operacionalização;
- ⇒ Documentar o *rationale* da escolha da alternativa para operacionalização;
- ⇒ Possibilitar o rastro dos requisitos de segurança de alto nível até sua operacionalização (como são refinados em requisitos de nível de abstração menor; como são operacionalizados);

⇒ Descrever os requisitos de segurança de uma forma que facilite sua validação.

O escopo desta dissertação está delimitado em requisitos de segurança (*security*). Não é objetivo desta dissertação abordar fidedignidade (*dependability*) nem seguridade (*safety*).

1.3 Trabalhos Relacionados

A elicitação, modelagem e análise de requisitos de segurança específicos da aplicação é uma área que vinha sendo pouco explorada pela engenharia de requisitos até recentemente, como afirma Lamsweerde em [10]. Entretanto, recentemente esta área vem recebendo mais atenção da comunidade científica com alguns trabalhos publicados nos últimos anos. Uma parte considerável dos trabalhos recentes nesta área pode ser agrupada em dois segmentos:

(i) abordagens baseadas em anti-requisitos, como *Security Use Cases* [11], *Eliciting security requirements by misuse cases* [12] e *Misuse Cases: Use Cases with Hostile Intent* [24];

(ii) abordagens baseadas em estratégias orientadas a metas para elaboração dos requisitos de segurança, como *Security and Privacy Requirements Analysis within a Social Setting* [3], *Security Design Based on Social Modelling* [37], *Elaborating security requirements by construction of intentional anti-models* [10], *Intentional Modeling to Support Identity Management* [16], *Handling Obstacles in Goal Oriented Requirements Engineering* [19], *Requirement Elicitation Based on Goals with Security and Privacy Policies in Electronic Commerce [36]* e *Security and Trust Requirements Engineering* [28].

Nas abordagens baseadas em anti-requisitos, a âncora são os anti-requisitos. Um *anti-requisito* é um requisito de um usuário malicioso que subverte um requisito existente [27]. Nestas abordagens, em linhas gerais, os anti-requisitos são derivados a partir dos requisitos da aplicação. Firesmith vai além, em [11], derivando os requisitos de segurança dos anti-requisitos.

Nas abordagens baseada em estratégias orientadas a metas, as intenções são a âncora. Para os anti-requisitos, as intenções maliciosas dos atacantes são a âncora. Para os requisitos de segurança, as intenções dos atores do sistema são a âncora. Nestas abordagens, em linhas gerais, os anti-requisitos são derivados como alternativas para satisfação das intenções maliciosas dos atacantes e os requisitos de segurança como contra medidas de defesa dos atores do sistema.

No capítulo 5, é apresentada uma análise mais rica de alguns dos trabalhos mais relevantes.

1.4 Conceitos

No contexto desta dissertação, serão utilizados os seguintes conceitos:

- ⇒ **Requisito Não-Funcional (NFR)**: em engenharia de software, um requisito de software que descreve não o que o software deve fazer, mas como o software irá fazer isso, por exemplo, requisitos de desempenho do software, requisitos de interface externa do software, restrições da arquitetura (*design constraints*) do software, e atributos de qualidade do software. Requisitos não-funcionais são difíceis de testar; consequentemente eles são geralmente avaliados subjetivamente [29].

- ⇒ **Requisito Funcional**: um requisito de software/sistema que especifica uma função que o software/sistema ou componente do software/sistema deve ser capaz de desempenhar. Estes são os requisitos de software que definem o comportamento do sistema, isto é, o processo fundamental ou transformação que os componentes de software e hardware desempenham em entradas para produzir saídas [29].

- ⇒ **Confidencialidade**: ausência de revelação (*disclosure*) não-autorizada de informação [14].

⇒ **Consistência**[1]: ausência de alterações impróprias (não-autorizadas) do sistema.

⇒ **Integridade**: composição dos atributos de exatidão, consistência e completeza [7].

⇒ **Segurança (*security*)**: composição dos atributos de confidencialidade, integridade e disponibilidade [14].

⇒ **Fidedignidade (*Dependability*)**: habilidade de evitar falhas de serviço que sejam mais freqüentes ou mais severas que o aceitável [14].

⇒ **Disponibilidade**: prontidão para o serviço correto. [14]

⇒ **Seguridade (*Safety*)**: ausência de conseqüências catastróficas para o(s) usuário(s) e para o ambiente [14].

⇒ **Serviço correto**: serviço (entregue) que implementa as funções do sistema [14].

⇒ **Operacionalização**: técnica de desenvolvimento usada na arquitetura (*design*) ou implementação do sistema alvo para ajudar a 'satisfazer a contento' os requisitos não-funcionais [7]. Em outras palavras, é uma técnica para desdobrar os requisitos não-funcionais em requisitos funcionais.

1.5 Organização da dissertação

Esta dissertação apresenta uma abordagem orientada a metas para definição (elicitação, modelagem e análise) de requisitos de segurança específicos de aplicação.

[1] Em [14], consistência (e não integridade) é definida como "a ausência de alterações impróprias (não-autorizadas) do sistema. Nós optamos por considerar, como Chung et al [7], consistência como um sub-tipo de integridade, com o mesmo significado dado a integridade em [14].

No capítulo 2 é apresentado um breve resumo do *framework* i* [1] (um *framework* de modelagem intencional) que usamos para modelar os requisitos em nosso trabalho. Neste capítulo também são apresentadas duas extensões ao *framework* original propostos por Yu, que usamos em nossa abordagem.

O capítulo 3 apresenta a abordagem proposta em nosso trabalho para definição dos requisitos de segurança, baseada em um trabalho prévio de Liu, Yu e Mylopoulos [3].

No capítulo 4, é apresentado um estudo de caso com o exercício, passo a passo, da abordagem proposta em nosso trabalho.

O capítulo 5 apresenta uma análise de alguns trabalhos relevantes com abordagens distintas para definição de requisitos de segurança, situando-os em relação ao nosso trabalho.

No capítulo 6 são apresentadas as conclusões e contribuições do nosso trabalho, além das expectativas em relação a trabalhos futuros.

2
O *framework* i*

Este capítulo apresenta o *framework* i*, no qual está baseada a abordagem orientada a metas usada nesta dissertação. Ao longo do capítulo será apresentada uma visão geral do *framework* i*, os modelos SD e SR e por fim, duas extensões do *framework* i* original: o modelo SA (*Strategic Actor*) e *SDsituations*.

2.1 Visão geral do *framework* i*

O *framework* i* (i-estrela) [1] modela contextos organizacionais baseado nos relacionamentos de dependência entre os atores. O i* é um *framewok* usado para obter um melhor entendimento dos relacionamentos organizacionais entre os vários agentes organizacionais. Os relacionamentos entre os atores ajudam os engenheiros de requisitos a elicitar metas nas fases iniciais (Por que os atores têm estas dependências?). O uso do *framework* i* possibilita a compreensão das razões internas dos atores, uma vez que as mesmas são expressas explicitamente, auxiliando na escolha de alternativas durante a etapa de modelagem do software.

No i*, os participantes do contexto organizacional são chamados de atores. De acordo com Yu [1], atores são entidades ativas que efetuam ações para alcançar metas através do exercício de suas habilidades e conhecimentos, podendo ter dependências intencionais entre si. Uma dependência intencional ocorre quando o elemento da dependência está, de alguma forma, relacionado a alguma meta de um dos atores.

O *framewok* i*, como originalmente proposto por Yu em [1], usa dois modelos: o modelo SD (*Strategic Dependency*) e o modelo SR (*Strategic Rationale*), descritos a seguir.

2.2 Modelo SD

O modelo SD é usado para mapear a rede de dependências entre os atores organizacionais. O modelo SD é composto por um conjunto de nós que representam os atores (agentes, posições ou papéis) e elos que representam as dependências entre os atores. Uma dependência representa um acordo entre dois atores, onde um ator (“*depender” ou “dependente”*) depende de um outro ator (“*dependee*” ou “*de quem se depende”*) para que uma meta (*goal*) seja alcançada, uma tarefa seja executada, um recurso seja disponibilizado ou uma meta-flexível (*softgoal*) seja razoavelmente satisfeita. Metas, metas flexíveis, tarefas e recursos que fazem parte das relações de dependência entre os atores são chamados de *elementos de dependência* (*dependum*) e caracterizam os quatro tipos de dependência possíveis, descritos a seguir:

- **Dependência por meta**: ocorre quando o *depender* depende do *dependee* para que certo estado do mundo seja alcançado. Ao *dependee* é dada a liberdade de escolher como fazê-lo. Com uma dependência por meta, o *depender* ganha a habilidade de assumir que a condição ou estado do mundo será alcançado, mas torna o *depender* vulnerável, pois o *dependee* pode falhar em realizar tal condição;

- **Dependência por tarefa**: ocorre quando um ator (o *depender*) depende de outro (o *dependee*) para que este outro desempenhe uma tarefa. Uma dependência por tarefa especifica “como” a tarefa deve ser desempenhada, mas não especifica “porquê”. O *depender* é vulnerável, pois o *dependee* pode falhar em desempenhar a tarefa. A especificação de uma tarefa deve ser vista mais como uma restrição do que como o conhecimento prévio adequado (*knowhow*) para realização da tarefa.

- **Dependência por recurso:** ocorre quando um ator (o *depender*) depende de outro (o *dependee*) para que uma entidade (física ou computacional) seja disponibilizada. Com o estabelecimento deste tipo de dependência, o *depender* ganha a habilidade de usar a entidade como um recurso, ficando ao mesmo tempo vulnerável, pois a entidade pode tornar-se indisponível.

- **Dependência por meta flexível:** ocorre quando um ator (o *depender*) depende de outro (o *dependee*) para que este desempenhe alguma tarefa para que uma meta flexível seja "satisfeita a contento" ou "razoavelmente satisfeita", isto é, seja satisfeita a um nível considerado aceitável. A expressão "satisfeita a contento" aqui é utilizada como uma tradução do termo em inglês *satisficy*, introduzido por Herbert Simon [2] nos anos 1950. O conceito de meta flexível é usado para designar metas cujos critérios para sua satisfação não estão claramente definidos *a priori,* enquanto o conceito de meta rígida, chamadas apenas metas, está baseado numa noção clara em relação a sua satisfação, do tipo sim ou não - metas (rígidas) são usadas para representar determinados estados em que o mundo está ou não. (Esta subjetividade quanto a sua satisfação é inerente às metas flexíveis, e aos requisitos não-funcionais - por se tratarem de requisitos de qualidade, os critérios para sua satisfação não estão claramente definidos *a priori*.) Numa dependência por meta flexível não há um acordo prévio entre os atores (*depender* e *dependee*) sobre o que constitui a satisfação da meta (flexível). O significado da meta flexível é especificado em termos dos métodos que são escolhidos no curso da busca para alcançar a meta. Assim com em uma dependência por meta, o *depender* ganha a habilidade de assumir que a condição ou estado do mundo será alcançado, mas torna-se vulnerável, pois o *dependee* pode falhar em realizar tal condição. Diferentemente do que ocorre com as metas, as condições para que as metas flexíveis sejam alcançadas são elaboradas ao passo que as tarefas são desempenhadas.

Estes quatro tipos de dependência também caracterizam como as decisões de processo (escolha de alternativas; seqüência de passos para execução de uma tarefa) recaem em cada lado da dependência. Em uma dependência por meta, o *dependee* tem liberdade para tomar as decisões necessárias para que a meta seja satisfeita. Em uma dependência por tarefa o *depender* toma as decisões. Em uma dependência por recurso a questão da decisão não vem à tona, por ser o recurso considerado um produto final de algum processo de ação-deliberação, não havendo assim nenhuma decisão em aberto a ser considerada. Em uma decisão por meta flexível, o *depender* toma a decisão final, mas faz isso beneficiado pelo

conhecimento prévio adequado (*knowhow*) do *dependee*. A Figura 2.1 ilustra como são representados os quatro tipos de relação em um modelo SD.

2.3 Modelo SR

O modelo SR tem como objetivo representar as estratégias internas de cada ator. O modelo SR provê uma descrição intencional do processo em termos dos elementos do processo e das decisões e escolhas por trás dele. O modelo SR permite representar explicitamente o entendimento detalhado do raciocínio (*rationale*) dos atores do processo. Estes modelos são elaborados de modo a expressarem o processo pelo qual: as metas são alcançadas, as tarefas são elaboradas, os recurso são disponibilizados e as metas flexíveis são refinadas (decompostas) e operacionalidadas. Isto é feito pela representação dos relacionamentos intencionais que são internos aos atores através de relacionamentos "meios-fim" que relacionam elementos de processo, provendo uma representação explícita do 'porque', do 'como' e de alternativas.

O modelo SR é um grafo, com alguns tipos de nós e elos que provêem uma estrutura de representação para expressar explicitamente as razões por trás do processo. Há quatro tipos de nós, idênticos aos tipos de *dependum* em um modelo SD: meta, tarefa, recurso e meta flexível. Há duas classes principais de elos: elos de decomposição de tarefa e elos "meios-fim".

Elos de decomposição de tarefa – Uma tarefa (cuja noção está associada a 'como fazer alguma coisa') é modelada em termos de sua decomposição em suas sub-componentes, que podem ser: metas, tarefas, recursos e/ou metas flexíveis (os quatro tipos de nós).

Uma **meta** é uma condição ou estados de desejos no mundo que o ator deseja alcançar, expressa como uma assertiva na linguagem de representação. Como a meta deve ser alcançada não é especificado, possibilitando a consideração de várias alternativas.

Uma **tarefa** especifica um modo particular de fazer alguma coisa. Quando uma tarefa é especificada como sub-tarefa (parte) de outra tarefa (todo) ela

restringe esta outra tarefa (todo) a um curso de ação em particular, especificado pela tarefa sub-tarefa.

Um **recurso** é uma entidade (física ou informacional) que não é considerada problemática pelo ator. A principal característica é se está disponível (e por quem foi disponibilizado no caso de uma dependência externa).

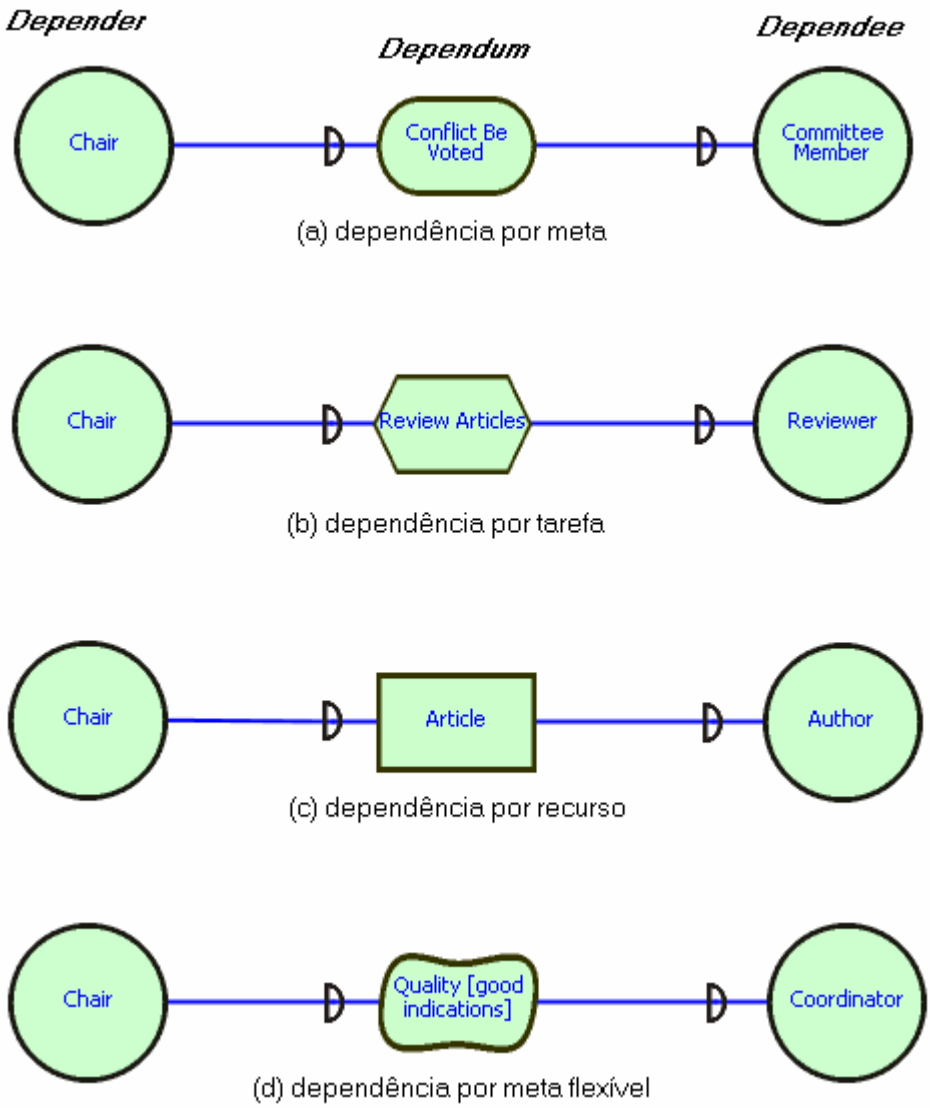

Figura 2.1 - Tipos de dependência

Uma **meta flexível** é uma condição ou estado no mundo que o ator deseja alcançar, mas diferentemente de uma meta (rígida), o critério para a condição ser alcançada não é precisamente definido *a priori*, estando sujeito à interpretação. Quando uma meta flexível é um componente em uma decomposição de tarefa, ela

serve como uma meta de qualidade para aquela tarefa, guiando (ou restringindo) a seleção entre as alternativas para a decomposição da tarefa.

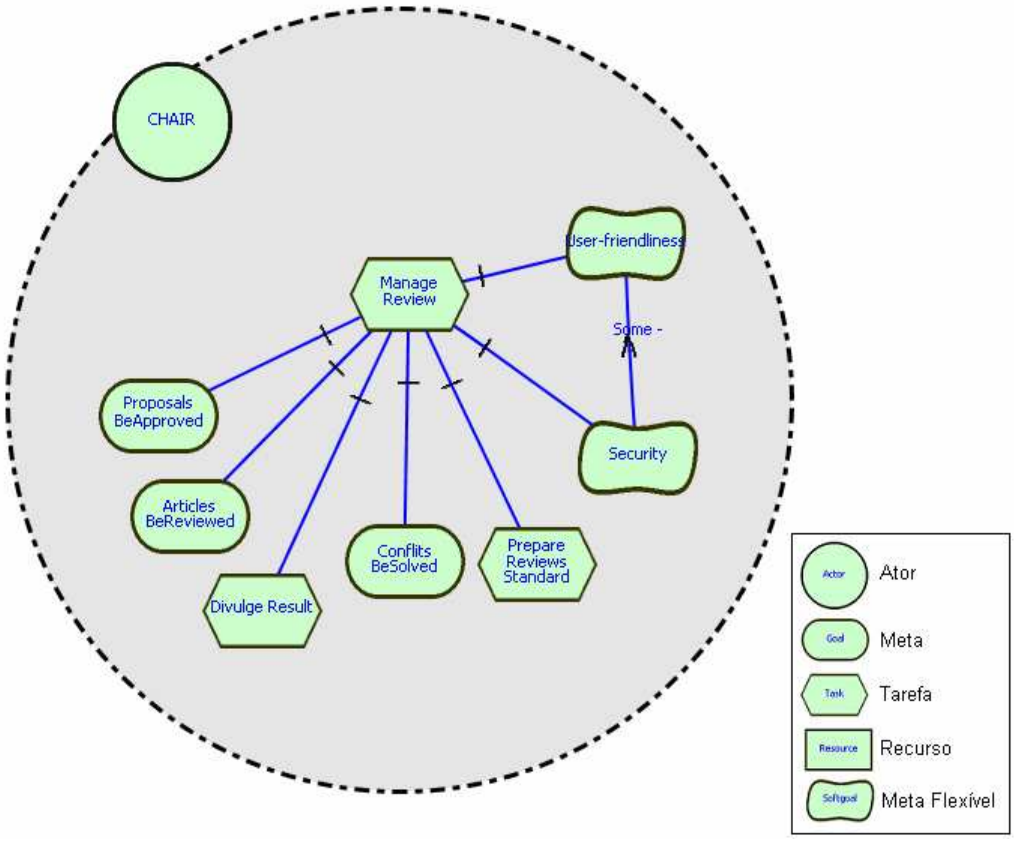

Figura 2.2 - Elos de decomposição de tarefa

No exemplo com elos de decomposição de tarefa apresentado na Figura 2.2, a tarefa “Manage Review” é decomposta em três (sub) metas: “Proposals Be Approved”, “Articles Be Reviewd” e “Conflicts Be Solved”; duas (sub) tarefas: “Divulge Result” e “Prepare Reviews Standard”; e duas metas flexíveis: “User-friendliness” e “Security”. A meta flexível de segurança contribui negativamente para a meta flexível de usabilidade, como indica o elo de contribuição “Some-” de “Security” para “User-friendliness”.

Elos Meios-Fim – Elos do tipo meios-fim indicam relacionamentos entre um “fim”, que pode ser: uma meta a ser alcançada, uma tarefa a ser desempenhada,

um recurso a ser produzido, ou uma meta flexível a ser razoavelmente satisfeita (*satisficed*); e "meios" alternativos para alcançá-lo. Os meios são usualmente expressos na forma de tarefas, uma vez que a noção de tarefa está associada a 'como fazer alguma coisa'. Na notação gráfica do i*, uma cabeça de seta aponta dos meios para o fim. Os tipos de elos meios-fim[2] estão descritos a seguir:

> Elo Tarefa-Meta – em um elo tarefa-meta o "fim" é especificado como uma meta e o "meio" é especificado como uma tarefa. Esta tarefa especifica "como" através de sua decomposição em seus componentes.
>
> Elo Tarefa-Recurso - neste tipo de elo o "fim" é especificado como um recurso e o "meio" é especificado como uma tarefa.
>
> Elo Tarefa-Meta Flexível - neste tipo de elo o "fim" é especificado como uma meta flexível e o "meio" é especificado como uma tarefa.
>
> Elo Meta Flexível–Meta Flexível – nos elos do tipo Meta Flexível–Meta Flexível tanto o "fim" quanto o "meio" são especificados como metas flexíveis. Este tipo de elo possibilita o desenvolvimento de uma hierarquia meios-fim de metas flexíveis, até que eventualmente algumas metas flexíveis são operacionalizadas por tarefas (via elos Tarefa-Meta Flexível). Esta hierarquia é particularmente interessante para o refinamento de metas flexíveis, onde uma meta flexível com nível de abstração alto é refinada sucessivamente até que se consigam metas flexíveis mais fáceis de operacionalizar.

Os elos meios-fim que envolvem metas flexíveis possuem um atributo extra para indicar o tipo de contribuição. A escala de contribuição usada por Yu em [1] é simplificada e apresenta três valores para contribuição: positiva ("+"), negativa ("-") e neutra ("?"), ao passo que Chung et al [7] usa uma escala de contribuição com cinco valores: *brake* ("--"), *hurt* ("-"), *unknown* ("?"), *help* ("+") e *make* ("++").

[2] Embora o tipo de elo se chame meios-fim (*means-end*), Yu [1] os descreve de forma invertida: fim-meios. Visando facilitar a compreensão, optamos por descrevê-los de forma direta, diferentemente de Yu.

Yu menciona ainda em [1, pág 35] a possibilidade de uso de outros tipos de elos meios-fim como meta-meta, embora este tipo de elo não apareça em nenhum dos exemplos apresentados.

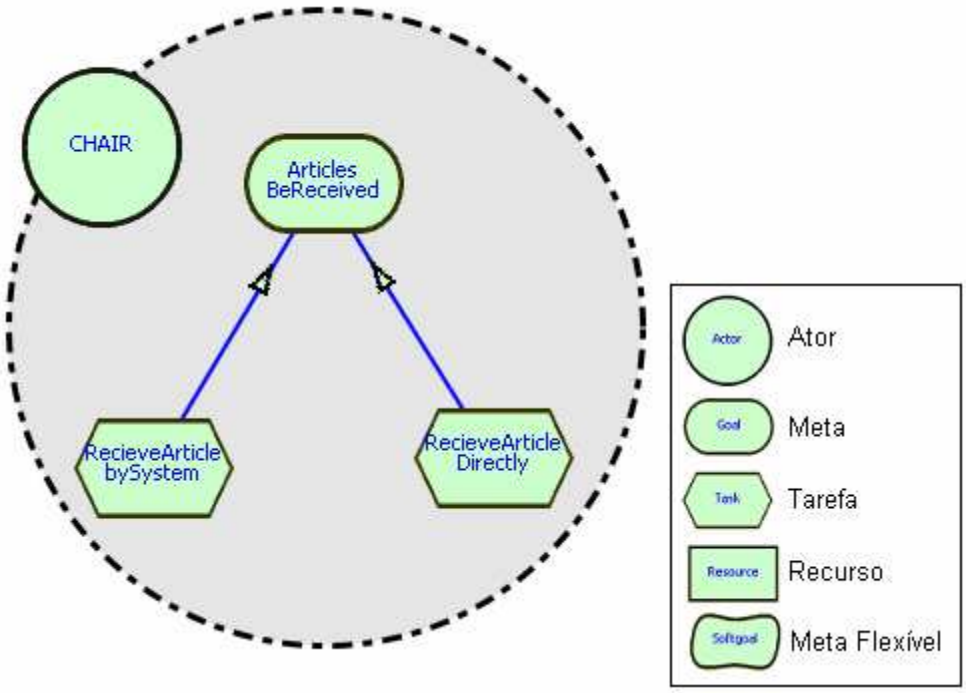

Figura 2.3 - Elos meios-fim

No exemplo de uso de elos meios-fim apresentado na Figura 2.3, há dois elos do tipo Meta-Tarefa, em ambos a meta (“fim”) é “ArticlesBeReceived” sendo as tarefas “ReceiveArticlebySystem” e “ReceiveArticleDirectly” os meios.

2.4 Extensões do *framework* i*

Duas extensões ao *framework* original proposto por Yu serão usadas neste trabalho: o modelo SA (Strategic Actor) e *SDSituations*. Estas extensões são detalhadas a seguir.

2.4.1 O Modelo SA (*Strategic Actor*)

O modelo SA [13] é usado especificamente para modelar os atores em i*. A adoção do modelo SA auxilia no entendimento dos atores, e seus relacionamentos, do ponto de vista estrutural. Embora Yu não tenha proposto um modelo específico com esta finalidade, o modelo SA usa os mesmos conceitos presentes em [1]: agente, posição e papel como refinamento dos atores; e os relacionamentos entre eles. Estes conceitos estão descritos a seguir, conforme definido por Yu [1]:

> **Ator:** "Um ator é uma entidade ativa que desempenha ações para alcançar suas metas através do uso de seu conhecimento" [1, pág 12]. "O termo ator é usado para se referir genericamente a qualquer unidade a qual se possa atribuir dependências intencionais." [1, pág. 17].
>
> **Posição:** "Uma posição está em um nível intermediário de abstração entre um papel e um agente. É um conjunto de papéis tipicamente desempenhado". [1, pág. 17].
>
> **Papel:** "Um papel é uma caracterização abstrata do comportamento de um ator social em algum contexto ou domínio especializado. Suas características podem facilmente ser transferidas para outros atores sociais. Dependências são associadas com papéis quando estas dependências se aplicam independentemente de quem esteja desempenhando o papel". [1, pág. 17].
>
> **Agente:** "Um agente é um ator com manifestações físicas, concretas, tal qual um ser humano. O termo agente é usado no lugar de pessoa a fim de generalizar, podendo se referir tanto a agentes humanos quanto a agentes artificiais (hardware/software). Um agente tem dependências que se aplicam independentemente de que papel ele está desempenhando. Estas características não podem ser tipicamente transferidas para outros indivíduos, como por exemplo, suas habilidades, suas experiências, suas limitações físicas". [1, pág. 17].

Em virtude do relacionamento de instanciação entre agentes, foi proposto em [13] um novo tipo de ator: o agente real. O agente real é mais específico que o

agente (mais genérico), podendo ser identificado unicamente, como por exemplo, uma pessoa específica ou um software específico.

Os relacionamentos entre estes conceitos conforme descritos por Leite et al. em [13], baseado nas definições e exemplos de [1], são apresentados resumidamente a seguir:

Ocupa - relacionamento de agente para posição – um agente pode ocupar mais de uma posição e uma posição pode ser ocupada por mais de um agente.

Desempenha – relacionamento de agente para papel – um agente pode desempenhar mais de um papel e um papel pode ser desempenhado por mais de um agente.

Cobre – relacionamento de posição para papel – uma posição pode cobrir mais de um papel e um papel pode ser coberto por mais de uma posição.

Parte de – relacionamento de posição para posição; de papel para papel; de agente para agente – posição, papel e agente podem ter mais de uma sub-parte, assim como podem ser sub-parte de mais de uma posição, papel e agente respectivamente. Há uma restrição para este relacionamento entre posições. Uma posição é parte de outra se todos os papéis desta posição também são cobertos pela outra posição.

É um – relacionamento de ator para ator; de posição para posição; de papel para papel; de agente para agente – ator, posição, papel e agente podem ser especializados por mais de um ator, posição, papel e agente respectivamente.

Instancia – relacionamento de agente real para agente – um agente real instancia um agente; um agente pode ser instanciado por mais de um agente real.

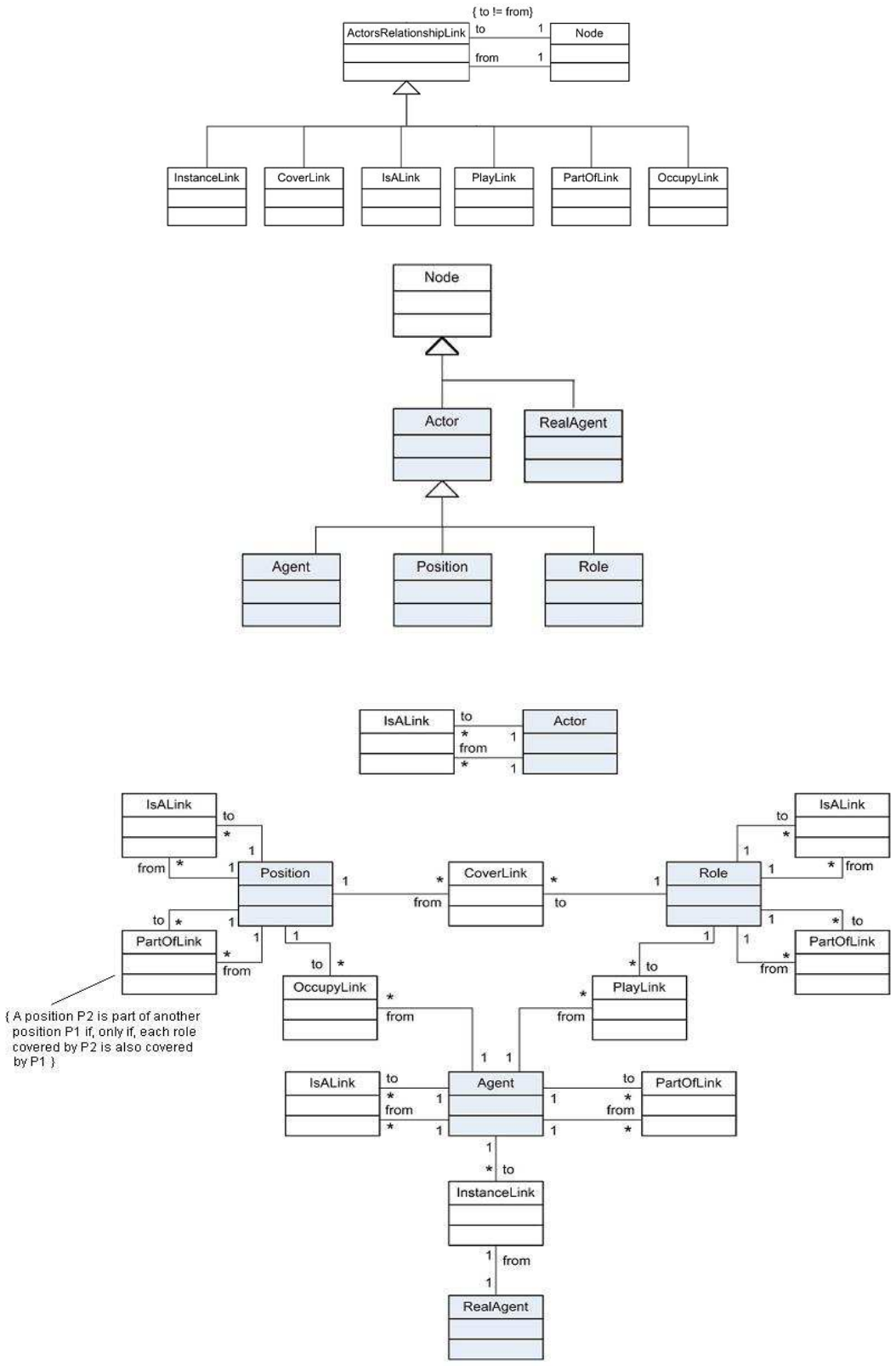

Figura 2.4 - Meta-modelo elaborado em UML para *Strategic Actor* (SA) proposto em [13]

Um modelo SA é um grafo cujos nós são derivados do refinamento de atores: posição, papel, agente e agente real; e cujos elos são os relacionamentos ocupa, desempenha, cobre, parte de, é um e instancia. A Figura 2.4 apresenta um meta-modelo elaborado em UML proposto em [13].

2.4.2 *SDsituations* (Situação de Dependência Estratégica)

SDsituation é uma forma de representação estruturada de uma situação de dependência estratégica, proposta por Oliveira e Cysneiros em [4]. As situações de dependência ocorrem naturalmente em ambientes organizacionais. A idéia central é que cada elo de dependência envolvendo atores não é isolado, e sim parte de uma situação bem definida de colaboração chamada de "*strategic dependency situation"* ou *SDsituation*.

Uma *SDsituation* é composta por um ou mais elementos de dependência. Cada *SDsituation* pode ser identificada separadamente das outras *SDsituations* formando uma cadeia de interdependências. Os elementos de dependência de uma *SDsituation* são os mesmos elementos de dependência (*dependum*) presentes em um modelo SD: meta, tarefa, recurso e meta flexível. As interdependências entre *SDsituations* podem ser de três tipos: física, lógica e temporal. Mais de um tipo de interdependência entre as *SDsituations* podem ocorrer ao mesmo tempo. Os três tipos de interdependências entre *SDsituations* estão descritos a seguir:

- Física - ocorre quando um recurso é preparado por uma *SDsituation* e é necessitado por outra *SDsituation*;

- Lógica - ocorre quando uma ou mais *SDsituations* necessitam da conclusão de outra *SDsituation* para sua iniciação ou quando uma ou mais *SDsituations* necessitam da conclusão de outra *SDsituation* para sua conclusão.

- Temporal – ocorre quando uma ou mais *SDsituations* necessitam esperar algum tempo após o início de outra *SDsituation* ou quando uma ou mais *SDsituations* necessitam esperar algum tempo após a conclusão de outra *SDsituation*.

Uma *SDsituation* é caracterizada principalmente pelo conjunto de elementos de dependência que fazem parte da situação de dependência. Além deste conjunto de elementos de dependência, fazem parte da definição de uma *SDsituation* o título, a meta (a meta principal da situação de dependência), e a descrição da situação de dependência. A definição de uma *SDsituation* é feita seguindo três passos básicos:

(i) identificação dos elementos da *SDsituation* - tanto os atores, *depender* e *dependee*, quanto os elementos de dependência entre eles, o *dependum* (metas, tarefas, recursos e metas flexiveis) podem ser identificados a partir do modelo SD. Caso tenha sido construído um Léxico Ampliado da Linguagem (LEL), este pode ser o ponto de partida para identificação dos elementos da *SDsituation* conforme sugerem Oliveira e Cysneiros em [4];

(ii) composição de *SDSituations* – uma *SDsituation* é composta por uma ou mais dependências. Examinando as dependências no modelo SD, o engenheiro de requisitos deve definir que dependências fazem parte de uma mesma situação;

(iii) identificação de interrelacionamento entre *SDsituations* – identificação de dependências (lógica, física ou temporal) entre as *SDsituations.* Apesar de uma *SDsituation* ser composta de uma ou mais dependências, uma dependência pode ser uma outra *SDsituation*. É necessário examinar as *SDsituations* identificando para cada uma as dependências de outras *SDsituations*, se existirem.

A seguir, é apresentada como exemplo a descrição da *SDsituation* ‘Revisar artigo’ (*Article Review*). As *SDsituations* serão apresentadas com mais detalhes no estudo de caso no capítulo 4.

SDsituation Definition				
SDsituation:	**ARTICLES REVIEW**			
Goal:	ArticlesBeReviewed			
Definition:	It represents a situation when the Reviewer reviews an article (the proposal of review has been previously accepted)			
ACTOR (depender)	ELEMENTS OF DEPENDENCY	TYPE	DEGREE	ACTOR (dependee)
	PROPOSAL ACCEPTANCE	DEPENDENCY	critical	
Chair	Review Article	task	critical	Reviewer
Chair	Quality [Good Review]	softgoal		Reviewer
Reviewer	Article to Review	resource	critical	Chair
Chair	Reviewed Article	resource	critical	Reviewer

3
Uso de estratégias orientadas a metas

A abordagem proposta para o tratamento dos requisitos de segurança está baseada na adoção de estratégias orientadas a metas para definição dos requisitos. Estratégias orientadas a metas caracterizam-se pela construção de modelos intencionais, cujo foco se encontra nas metas a serem alcançadas pelos atores, ou seja, em suas intenções e motivações. A aplicação destas estratégias orientadas a metas foi baseada na utilização do *framework* i*, apresentado anteriormente, cuja intencionalidade é distribuída entre os atores. Neste capítulo, será apresentado (a título de introdução e de forma bastante resumida) o processo de engenharia de requisitos. Em seguida, apresentamos o processo proposto por Liu, Yu e Mylopoulos [3]. Depois apresentamos a evolução do processo proposta por nós, detalhando os passos para definição de requisitos gerais e os passos específicos para definição de requisitos de segurança.

3.1 Processo de Engenharia de Requisitos

De acordo com Leite [30], o processo de engenharia de requisitos é composto de quatro macro-atividades: elicitação, modelagem, análise e gerência, conforme apresentado na Figura 3.1. A atividade de elicitação tem como principal objetivo a aquisição de informações (e requisitos), oriundas do "Universo de Informações". A atividade de modelagem tem por objetivo documentar os requisitos através de modelos. A atividade de análise tem por objetivo a verificação e validação dos requisitos modelados. A atividade de gerência é desempenhada paralelamente a estas três atividades, com o objetivo de evoluir os requisitos de forma controlada e garantir a rastreabilidade entre os requisitos.

Nesta dissertação, chamaremos de “Definição de Requisitos” a composição das atividades de elicitação, modelagem e análise, conforme indicado na Figura 3.1.

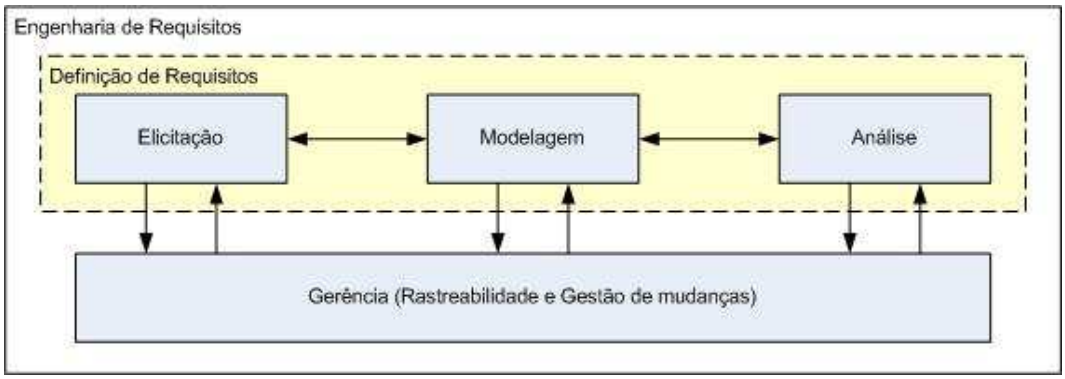

Figura 3.1 - Visão geral da Engenharia de Requisitos, adaptada de [30]

3.2 Definição de requisitos usando i*

Neste trabalho será utilizada uma extensão do processo de definição de requisitos usando i*, originalmente elaborado por Liu, Yu e Mylopoulos [3]. O processo tem por objetivo incorporar requisitos de segurança (não-funcionais) à definição de requisitos gerais, conforme representado na Figura 3.2.

O processo original é composto por duas partes: (i) um processo básico de definição de requisitos gerais (representado pelas caixas com linhas sólidas, à esquerda) e (ii) um processo para definição de requisitos específicos de segurança (representado pelas caixas com linhas tracejadas, à direita). O processo de definição de requisitos gerais apresenta os passos para construção e refinamento dos modelos SD e SR do *framework* i*. A este processo, são integrados os passos referentes ao processo de definição de requisitos específicos de segurança.

3.3 Evolução do processo de definição de requisitos

Nosso trabalho propõe a evolução do processo proposto por Liu, Yu e Mylopoulos em [3] com a incorporação de alguns passos ao processo de definição

de requisitos gerais. Estes passos foram introduzidos com os objetivos de: ajudar a lidar com a complexidade (para análise) dos modelos produzidos pelo *framework* i*; melhorar o entendimento dos atores e suas relações; tornar explícito o processo de definição dos requisitos não-funcionais, em especial, os de segurança; e, auxiliar a validação dos requisitos pelos usuários e demais interessados.

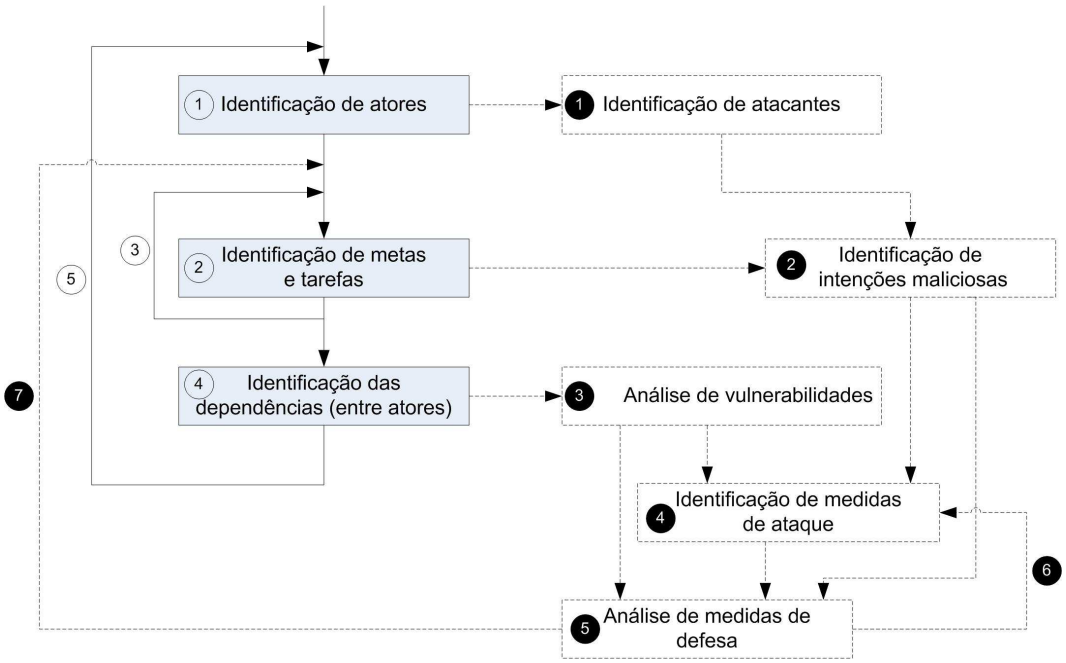

Figura 3.2 - Processo de definição de requisitos usando i* de Liu, Yu e Mylopoulos [3]

Os modelos SR produzidos em i* são grafos com vários tipos de nós (ator e seus refinamentos, meta, tarefa, recurso e meta flexível) e dois tipos de elo (meios-fim e decomposição de tarefa). Freqüentemente estes modelos tornam-se complexos (para análise): algumas tarefas alternativas para se alcançar uma mesma meta, detalhamento das tarefas alternativas através de sua decomposição. A adição dos requisitos de segurança torna estes modelos ainda mais complexos (para análise). Por este motivo foi introduzido um passo para definição de *SDSituations*. Para auxiliar o entendimento dos atores e suas relações foi introduzido um passo específico para refinamento dos atores, com elaboração de modelos SA (*Strategic Actor*). No processo original [3], os atores são modelados a partir dos passos identificação de atores e identificação de atacantes. Porém, os

atores (refinados ou não em posições, papéis e agentes) aparecem diretamente nos modelos SD e SR, elaborados nos passos subseqüentes. Com a introdução, feita por nós, de um passo específico para elaboração dos modelos SA, nossa intenção é que seja despendida uma atenção especial aos atores e suas relações durante a elaboração destes modelos, que servirão de referência (dos atores e relações) para o restante do trabalho.

A definição de cenários foi introduzida com o objetivo de auxiliar a validação dos requisitos. Por se tratar de uma estrutura descritiva, o uso de cenários prescinde do entendimento da semântica de uma notação gráfica rica como a de i* pelos atores validadores dos requisitos.

Os requisitos não-funcionais, particularmente os de segurança, precisam ser detalhados a um nível adequado para que análise de alternativas de operacionalização se torne viável. Para abordar esta questão, foi introduzido um passo para definição de requisitos não-funcionais.

As alterações introduzidas estão descritas a seguir:

(i) Definição de *SDsituations*

Conforme mencionado anteriormente, uma *SDsituation* é uma forma de representação estruturada de uma situação de dependência estratégica [4]. Uma *SDsituation* é composta de um ou mais elementos de dependência, e cada *SDSituation* pode ser identificada separadamente das outras, formando uma rede de interdependências. A interdependência entre *SDsituaions* pode ser física, lógica ou temporal. O uso de *SDsituations* ajuda a lidar com a complexidade inerente aos modelos de interdependências intencionais [4], sendo este o principal motivo para a sua incorporação ao processo de análise de requisitos. De fato, é mais fácil trabalhar com *SDsituations* do que com todas as dependências ao mesmo tempo, como pôde ser verificado no estudo de caso, apresentado mais adiante.

(ii) Definição de Cenários

Um cenário é uma descrição estruturada de uma situação que ocorre no mundo real [5]. Situações têm algumas características: são concretas e possuem metas ou objetivos; envolvem atores e necessitam de recursos; são individualmente independentes e inter-relacionadas; acontecem em um local e tempo definidos e podem ter restrições (uma restrição pode ao mesmo tempo qualificar ou designar alguma restrição ao cenário e é de certa maneira um requisito não funcional [8]). Um cenário satisfaz uma meta ou objetivo e tem um contexto que especifica suas fronteiras. Cenários são compostos por episódios que definem seus requisitos funcionais. Pode haver cursos alternativos de execução, representados por exceções, que o cenário deva considerar. O principal motivo para introdução da definição de cenários, ao processo de definição de requisitos gerais, é que são estruturas descritivas e, por isso, são mais adequados para a interação com os usuários finais, que se sentem mais confortáveis em validar um artefato descrito em linguagem natural. Um cenário usa a linguagem do "Universo de Informações". Esta linguagem é capturada através do Léxico Estendido da Linguagem (LEL) [6] que é um modelo de representação de seus termos, composto por símbolos (verbo, objeto, sujeito e estado) que representam palavras ou sentenças. Cada símbolo é identificado por um nome, ou nomes no caso de haver sinônimos, e representado por duas descrições: a primeira, chamada noção, é a denotação do símbolo, equivalente a uma descrição encontrada no dicionário; a segunda, chamada de impacto ou respostas comportamentais, é a conotação do símbolo que descreve a contextualização do símbolo no "Universo de Informações".

(iii) Refinamento de Atores

A análise dos atores e dos atacantes é um ponto fundamental para questões de segurança. Este passo tem por objetivo a construção dos modelos SA (vide seção 2.4.1) dos atores legítimos e dos atacantes.

Com estes modelos os relacionamentos estruturais (estáticos) entre os atores ficam evidenciados. Os relacionamentos do tipo ‘desempenha’ e ‘ocupa’ são de especial interesse para análise de segurança.

(iv) Definição de Requisitos Não-Funcionais (NFRs – *Non-Functional Requirements*)

A incorporação de um passo específico para Definição de Requisitos Não-Funcionais tem por objetivo detalhar uma forma de definição destes requisitos. Algumas questões relevantes relativas aos requisitos não-funcionais precisam de respostas explícitas. Exemplos destas questões são: “Qual a origem dos requisitos não-funcionais?”; “São requisitos de quais atores?”; “São requisitos do sistema como um todo ou cada cenário possui seus próprios requisitos não-funcionais?”; “Qual a relação entre os requisitos não-funcionais? Por exemplo: qual o impacto de requisitos de segurança sobre os requisitos de usabilidade?”. Uma outra vantagem da introdução deste passo é representar separadamente o requisito não-funcional das possíveis soluções a serem adotadas para atendê-lo. Por serem requisitos de qualidade, os requisitos não-funcionais são representados por metas flexíveis no *framework* i*, aplicando-se a eles o conceito de *satisficy* (assim como para as metas flexíveis, não se pode afirmar que os requisitos não-funcionais são plenamente satisfeitos; pode-se afirmar apenas que são “satisfeitos em um nível considerado adequado”). A elicitação dos requisitos não-funcionais será discutida detalhadamente adiante, na descrição do processo.

Além da introdução destes novos passos, foi feita uma troca na ordem de execução entre o passo 2 – “Identificação de metas e tarefas” e o passo 4 – “Identificação das dependências (entre atores)”. Esta troca se mostrou necessária para introdução dos novos passos, além de não trazer nenhum prejuízo ao processo (os modelos SD e SR são mutuamente complementares e, em [1], não há uma recomendação sobre a ordem seqüencial para elaboração destes modelos).

A evolução do processo, a partir da introdução destas alterações no processo original (Figura 3.2), gerou um novo fluxograma de processo no qual está baseado o estudo de caso apresentado neste trabalho, apresentado na Figura 3.3. As caixas com linhas sólidas, à esquerda na Figura 3.3, indicam os passos da definição de requisitos de gerais. As caixas sombreadas indicam os passos acrescentados ao processo original. As setas em azul foram acrescentas em virtude da inserção dos novos passos. Os passos do processo de definição de requisitos gerais são detalhados na seção 3.4.1, a seguir, enquanto os passos do processo de definição de requisitos específicos de segurança são detalhados mais adiante, na seção 3.4.2 deste capítulo.

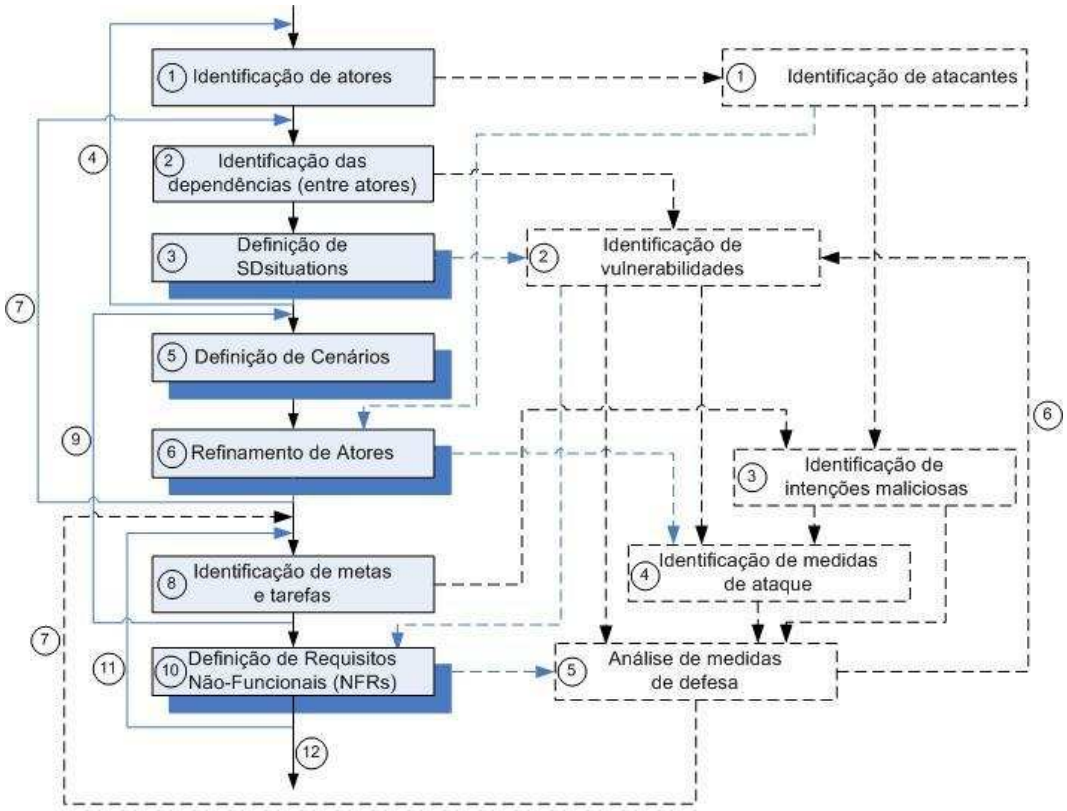

Figura 3.3 - Fluxograma do processo evoluído a partir do processo de Liu, Yu e Mylopoulos [3]

3.4 Detalhamento do processo evoluído

3.4.1 Detalhamento do processo evoluído de definição de requisitos gerais

Os passos do processo de definição de requisitos de gerais (caixas em azul, à esquerda na Figura 3.3) estão descritos detalhadamente a seguir:

① Identificação de atores – este passo responde a questão “quem está envolvido no sistema?”. Inicialmente são identificados atores humanos, representando os *stakeholders*, juntamente com os atores computacionais. Com o progresso da definição de requisitos, à medida que forem sendo tomadas algumas decisões de projeto durante a modelagem, mais atores serão identificados (passo 7 – Figura 3.3) como agentes de software, por exemplo. Este passo, identificação de atores, situa-se principalmente na atividade de elicitação de requisitos.

Em [3], neste passo já é feita o refinamento dos atores em agentes, papéis e posições. Nós preferimos postergar este refinamento para o passo 6 – refinamento de atores, após serem elaborados os modelos SD e definidas as *SDsituations* e cenários. Assim, há mais subsídios para o refinamento dos atores.

② Identificação das dependências entre atores – este passo responde a questão “como os atores se relacionam uns com os outros?”. As respostas a esta questão são representadas em um modelo SD. No *framework* i* o foco é o relacionamento intencional (ex: um ator depende de outro para que uma meta seja satisfeita) e não no fluxo de informações (ex: que mensagem um ator manda para outro) entre os atores. Este passo inicia-se na atividade de elicitação de requisitos, com a identificação das dependências, avançando até a fase de modelagem de requisitos, com a elaboração do modelo SD.

③ Definição de *SDsituations* – o objetivo deste passo é definir as situações de dependências entre os atores. Esta definição é feita da seguinte forma [4]:

(i) identificação dos elementos da *SDsituation* - identificar atores (*depender* e *dependee* podem ser identificados a partir do modelo SD). Os elementos de dependência correspondem aos *dependums* (metas, tarefas, recursos e metas flexiveis) no modelo SD;

(ii) composição de *SDSituations* – Examinando as dependências no modelo SD, o engenheiro de requisitos deve definir que

dependências (*depender* + *dependum* + *dependee*) entre os atores fazem parte de uma mesma situação de dependência estratégica (*SDsituation*). Estas dependências são agrupadas formando uma *SDsituation*;

(iii) identificação de interrelacionamento entre *SDsituations* – identificação de dependências (lógica, física ou temporal) entre as *SDsituations.* Apesar de uma *SDsituation* ser composta de uma ou mais dependências, uma dependência pode ser uma outra *SDsituation.*

Este passo situa-se na fase de modelagem de requisitos.

④ Passo iterativo: identificação de novos atores – após a identificação das situações de dependência entre os atores, novos atores podem ser identificados. Para tanto, deve-se examinar todas as dependência presentes em cada situação verificando se o *dependee* pode, por si só ou com a colaboração dos outros atores já identificados, 'honrar' os acordos estabelecidos com o *depender* representados pelos relacionamentos de dependência estratégica. Ou seja, deve se verificar se o *dependee* pode, sozinho ou com a colaboração dos demais atores identificados, satisfazer as metas, desempenhar as tarefas, disponibilizar os recurso ou 'satisfazer a contento' as metas flexíveis acordadas. A impossibilidade de 'honrar' os acordos indica a necessidade de novos atores, que devem ser identificados, que colaborem para o cumprimento das metas acordadas. Uma vez identificados estes atores, a colaboração será estabelecida via relacionamento de dependência estratégica, com a complementação do modelo SD elaborado previamente, de forma iterativa. Este passo reflete uma iteração entre as ativades de modelagem e elicitação de requisitos.

⑤ Definição de cenários – o objetivo deste passo é definir os cenários correspondentes às situações de dependência no modelo SD. A seguir são apresentados os passos para definição de cenários, juntamente com algumas heurísticas, propostos em [4] e [17], com adaptações:

(i) Criação dos cenários: embora seja possível que um cenário tenha mais de uma *SDsituation*, o usual é ter um cenário para cada

SDsituation;

(ii) Descrição dos cenários: descrever as informações do cenário, a saber:

a) Objetivo (ou meta) - considerar o melhor elemento de dependência da situação. A primeira escolha é uma dependência de meta, a segunda de tarefa e por último uma dependência de recurso. Uma dependência de meta flexível não pode ser o objetivo do cenário por que uma meta flexível está sempre junto de outro elemento de dependência;

b) Atores - os atores do cenário são os mesmos atores da *SDsituation*;

c) Contexto – esta informação não existe em *SDsituation*, mas pode ser incluído na descrição do cenário pelo engenheiro de requisitos;

d) Pré-condições - devem ser listados todos os recursos preparados em *SDsituations* anteriores e necessários à *SDsituation* corrente;

e) Episódios – descrição funcional dos detalhes do cenário, em forma de sentença. As sentenças descrevem as tarefas desempenhadas pelos atores no cenário. Algumas tarefas podem aparecer dentro de uma dependência de tarefa em uma *SDsituation,* derivadas do modelo SD, outras aparecerão nos modelos SD dos atores (elaborados no próximo passo). Com o refinamento dos cenários no passo 6, mais tarefas poderão ser acrescentadas aos episódios;

f) Restrições – designam metas flexíveis (NFRs) que devem ser consideradas e descrevem imposições do processo;

g) Exceções – representam alternativas na definição do cenário.

Na primeira execução deste passo, serão definidos cenários incompletos. Com a execução do passo 9 (iterativo), os cenários serão complementados.
⑥ Refinamento de atores – neste passo é construído um modelo SA. O foco do modelo SA são as relações estruturais entres os atores. Para tanto, os atores identificados anteriormente nos passos 1 e 2 são refinados em papéis, posições e agentes no modelo SA. Algumas heurísticas são usadas para construção do SA: primeiramente, atores que possuam características individuais relevantes para o contexto da aplicação e que não possam ser transferidas para outros atores devem ser desdobrados em agentes; para os demais atores no modelo SD, são criados papéis que representem a participação do ator nas *SDSituations*. Para as *SDSituations* complementares (com dependência lógica), pode ser criado apenas um papel. Para os atores que foram desdobrados em mais de um papel, pode se criar uma posição que ocupe estes papéis. Este refinamento de atores aplica-se também aos atacantes identificados no passo 1 (definição de requisitos específicos de segurança).
⑦ Passo iterativo: refinamento de modelos SD, cenários e modelos SR – com o progresso da definição de requisitos os atores já identificados podem ser refinados em papéis, posições ou agentes. Os modelos SD elaborados anteriormente, passo 2, são refinados com base nos modelos SA elaborados no passo 6.
⑧ Identificação de metas e tarefas – este passo responde a questão “o que os atores desejam alcançar?”. As respostas a esta questão podem ser representadas por metas que capturam os objetivos de alto-nível dos atores. Para estas metas são atribuídas tarefas alternativas, através de elos meios-fim, cuja execução possibilite o alcance das metas. Tarefas, por sua vez, podem ser decompostas em sub-tarefas, recursos, submetas (metas cujo alcance seja necessário para execução da tarefa) e metas flexíveis (que definem os requisitos de qualidade para execução da tarefa).
⑨ Passo iterativo: refinamento de cenários – o refinamento das metas, das metas flexíveis e das tarefas deve prosseguir até alcançar um estágio em que as decisões de desenho (*design*) possam ser tomadas com base nas informações

contidas nos modelos e cenários. À medida que as metas e tarefas são refinadas os episódios dos cenários correspondentes são enriquecidos, complementando a definição dos cenários.

⑩ Definição de requisitos não-funcionais (NFRs) – este passo tem por objetivo integrar os requisitos não-funcionais detalhados, relativos a cada *SDsituation*, aos modelos SR de cada ator, com os interrelacionamentos inerentes a estes requisitos. A abordagem é a seguinte:

(i) Criação de meta flexíveis –.É extremamente recomendado o uso de catálogos de NFRs, como proposto em [7] e [20], para a elicitação dos requisitos não-funcionais (metas flexíveis). Catálogos são uma forma interessante de se armazenar e reusar conhecimento a respeito dos requisitos não-funcionais: como estão inter-relacionados, quais as soluções conhecidas e que impacto estas soluções tem em outros requisitos. Em geral, o uso de um Catálogo de NFRs requer a adequação destes NFRs ao contexto da aplicação. Esta adaptação pode ser feita via metódos de refinamento do próprio catálogo [7]. As soluções encontradas para um determinado contexto poderão ser adicionadas ao catálogo para uso posterior;

(ii) Decomposição de requisitos não-funcionais – uma vez elicitados os requisitos não-funcionais, é necessário refiná-los até que alcancem um baixo nível de abstração em que seja possível identificar alternativas para sua operacionalização. Este refinamento é elaborado via métodos de decomposição [7]. Os requisitos de alto nível de abstração são decompostos primeiramente por tipos e, uma vez decompostos por tipos, são novamente decompostos por tópicos. Em uma decomposição por tipos, os requisitos não-funcionais de nível de abstração mais alto são decompostos em requisitos de características mais específicas. Em uma decomposição por tópicos, ocorre uma parametrização dos requistos não-funcionais por tópicos (“um *tópico* de uma meta flexível pode se referir a uma coleção de

itens, tais como informações, processos, funções, eventos, interfaces, tempo, quantia monetária, etc.” [7]).

A decomposição dos requisitos não-funcionias, tanto por tipos quanto por tópicos, quebra um requisito “pai” (de nível de abstração mais alto) em requisitos “filhos” usando elos de contribuição “E” (*AND*). Os métodos de decomposição possibilitam o exercício do paradigma “dividir para conquistar” [7].

(11) Passo iterativo: adição dos requisitos não funcionais aos modelos SR – Neste passo, os requisitos não-funcionais definidos no passo 10 são adicionados aos modelos SR, elaborados no passo 8. A execução destes passos deve ocorrer de forma iterativa até que os modelos estejam refinados o suficiente para que seja possível tomar decisões de desenho (*design*).

(12) Verificação de consistência entre modelos (passo final): ao final do processo é necessário verificar a consistência entre os modelos produzidos, como condição de parada para execução do processo. Caso os modelos ainda não estejam consistentes entre si, devem ser executados os passo iterativos 11, 9, 7 e 4.

3.4.2 Detalhamento do processo evoluído de definição de requisitos gerais

As caixas com linhas tracejadas, à direita da Figura 3.3, indicam os passos do processo de definição de requisitos específicos de segurança. Estes passos estão integrados ao **processo de definição de requisitos gerais** de forma a permitir que sejam antecipadas ameaças de atacantes potenciais e que sejam procuradas e adotadas contramedidas para proteção do sistema. Os passos relativos à definição de requisitos específicos de segurança estão descritos a seguir:

❶ Identificação de atacantes – este passo tem por objetivo identificar potenciais atacantes ao sistema. A premissa básica é que todos são considerados

"culpados até que provem sua inocência". Em outras palavras, cada um dos os atores (agentes, papéis ou posições) identificados no processo de definição de requisitos gerais podem ser considerados um potencial atacante ao sistema ou aos outros atores. Estes atacantes herdam as intenções, capacidades e relacionamentos sociais dos atores legítimos correspondentes, isto é, a hierarquia interna de metas e os relacionamentos de dependência externa no modelo. O termo atacante é usado para referenciar a fonte de qualquer ameaça. Uma fronteira é delineada para o sistema e, então, é feita uma busca exaustiva por atacantes dentro desta fronteira. Nesta abordagem, todos os atacantes são considerados atacantes "internos". Além dos "atacantes internos" derivados dos atores legítimos, identificados à esquerda da Figura 3.3 no passo 1 - Identificação de Atores, outros atacantes podem ser adicionados ao modelo. A identificação destes outros potenciais atacantes, que chamaremos de "atacantes externos", pode ser derivada das respostas para a pergunta: "Quem poderia se beneficiar de um ataque (bem-sucedido) à segurança do sistema?". O exame de informações históricas de ataques a aplicações do mesmo domínio, ou aplicações que usem o mesmo tipo de tecnologia, também pode auxiliar na identificação dos atacantes externos. Neste contexto, atacantes aleatórios como *hackers/crackers* da internet são representados como "atacantes externos" compartilhando o mesmo território que suas vítimas. Este passo é parte da atividade de elicitação de requisitos (específicos de segurança).

❷ Identificação de vulnerabilidades – Este passo visa identificar pontos de vulnerabilidade na rede de dependências (uma **vulnerabilidade** configura um risco [38, p 238] para o sistema e pode vir a ser explorado por um atacante, ou mesmo vir a ocorrer acidentalmente, sem intenção). A idéia básica é que relacionamentos de dependência trazem vulnerabilidades para o sistema e para o ator dependente (*depender*). Os elementos de dependência, presentes nos relacionamentos de dependência entre os atores, fazem parte da *superfície de ataque* do sistema (*superfície de ataque* de um sistema é definida em termos das ações que são visíveis externamente para seus usuários e dos recursos do sistema que ação acessa ou modifica [9]). Os relacionamentos de dependência constituem um bom ponto de partida para a identificação de vulnerabilidades. Atacantes em potencial podem explorar estas vulnerabilidades para atacar o

sistema e satisfazer suas intenções maliciosas. As questões a serem respondidas neste passo são: “Que relações de dependência são vulneráveis a ataques?”, “Qual o efeito (em cascata) se um relacionamento de dependência for comprometido?”.

A identificação de vulnerabilidades não se encerra com a identificação dos pontos de vulnerabilidade. É necessário examinar toda rede de dependências e verificar onde os relacionamentos de dependência atacados impactariam outros atores. O exame da rede de dependências pode ser iniciado a partir das *SDsituations*, identificando-se quais elementos de dependência de cada *SDsituation* são críticos em relação a determinados requisitos de segurança - uma vez determinado um ponto de vulnerabilidade, devem ser examinadas as outras *SDsituations* relacionadas (que possuam alguma dependência com a *SDstiuation* em exame) para avaliar o impacto da vulnerabilidade detectada. A modelagem de dependências em i* permite uma identificação de vulnerabilidades mais detalhada porque a falha potencial de uma dependência pode ser rastreada para o *depender* e para o *dependee* envolvidos. As vulnerabilidades identificadas nas redes de dependência (e nas respectivas *SDsituations*) sinalizam a necessidade de se adicionar metas flexíveis aos modelos SR dos atores envolvidos na dependência. É possível também identificar vulnerabilidades na infra-estrutura usada pelo sistema se recursos como internet, sistema operacional e outros softwares básicos forem tratados como recursos em i*. Este passo é parte da atividade de elicitação de requisitos (específicos de segurança).

3 Identificação de intenções maliciosas – Para cada atacante, identificado no passo 1 – identificação de atacantes, são identificadas, ou atribuídas, metas de alto nível que os atacantes desejem satisfazer ao atacar o sistema. Esta análise pode revelar a apropriação de recursos e capacidades legítimas para uso ilícito. Através do uso ilícito de recursos ou do exercício ilícito de ações do sistema o atacante visa se beneficiar de alguma forma. As questões chave deste passo são: “Que objetivos os atacantes desejam satisfazer ao atacar o sistema?”; “De que formas um atacante poderia se beneficiar se, ilicitamente, desempenhasse as ações ou acessasse os recursos do sistema (se possuísse as capacidades de um

ator real)?" Para responder a esta questão as relações de dependência (nos modelos SD) e as capacidades/habilidades dos atores legítimos (nos modelos SR) devem ser examinadas em busca do benefício potencial que um atacante poderia obter apropriando-se ilicitamente de recursos valiosos ou desempenhando ilicitamente as tarefas atribuídas aos atores legítimos. Durante a execução deste passo são elaborados modelos SR dos atacantes com as intenções maliciosas dos atacantes modeladas como metas. Este passo é parte da atividade de elicitação (identificação de intenções maliciosas) e também da atividade de modelagem de requisitos, com a elaboração de modelos intencionais dos atacantes.

4 Identificação de medidas de ataque – As medidas de ataque configuram ameaças à segurança do sistema. Este passo foca em como um atacante pode atacar os pontos de vulnerabilidade, identificados anteriormente, a fim de alcançar suas intenções maliciosas. O objetivo é identificar as diferentes medidas de ataque alternativas (meios) que possibilitem aos atacantes satisfazerem suas intenções maliciosas (fins). A pergunta chave deste passo é: "Quais as alternativas para satisfazer as intenções maliciosas?". Durante a execução deste passo, os modelos SR dos atacantes, elaborados no passo anterior, são refinados com as alternativas de ataque modeladas como tarefas que se relacionam com as intenções maliciosas através de elos "meios-fim". As estratégias identificadas nestes modelos SR configuram as medidas de ataque. Este passo é parte da atividade de elicitação (identificação de intenções maliciosas) e também da atividade de modelagem de requisitos, com o refinamento dos modelos intencionais dos atacantes produzidos no passo anterior.

5 Análise de medidas de defesa – Este passo consiste em identificar e analisar as alternativas de defesa para as ameaças previamente identificadas (via medidas de ataque). São fatores necessários para o sucesso de um ataque: a motivação do atacante, as vulnerabilidades do sistema e a capacidade do atacante de efetuar o ataque. Para neutralizar um ataque hipotético, é preciso encontrar medidas que neguem suficientemente estes fatores, eliminando a possibilidade da ocorrência de ataques ou reduzindo o impacto destes ataques

caso venham a ocorrer.

As medidas de defesa correspondem às alternativas de operacionalização para os requisitos não-funcionais de segurança refinados. Alternativas de operacionalização, representadas por tarefas, são relacionadas via elos de contribuição (meios-fim) que representam um refinamento "OU" (*OR*) às metas flexíveis (requisitos não-funcionais). Em geral, estes elos de contribuição possuem como atributo o valor de contribuição (que pode ser positivo ou negativo).

As alternativas de operacionalização também podem constar de um catálogo de requisitos não-funcionais, uma vez que podem ser reaproveitadas, embora com potencial de reuso menor que os requisitos refinados por tipos.

A ídeia é que sejam consideradas diversas alternativas de operacionalização. Uma vez que as diversas alternativas estejam explicitadas, com as respectivas contribuições (positivas e negativas) para os demais requitos não-funcionais, a escolha deve ser feita de acordo com a prioridade dos requistos impactados.

Através da avaliação do efeito das medidas de defesa contra as ameaças é possível decidir se os impactos das ameaças foram reduzidos a um nível aceitável. Esta análise é feita iterativamente até que uma solução satisfatória seja encontrada.

Um ponto relevante é que uma vulnerabilidade pode trazer danos a um sistema sem que haja necessariamente um ataque (no sentido de haver um atacante disposto a explorar uma vulnerabilidade do sistema em proveito próprio). Por ser i* um *framework* intencional, esta questão não é diretamente abordada neste processo. Para contornar esta omissão, a análise de medidas de defesa deve considerar não apenas as vulnerabilidades do sistema passíveis de ataque (intencionais), mas também vulnerabilidades às faltas não-intencionais, como acidentes, desastres naturais, mau-funcionamento de hardware e/ou software. Identificadas as medidas de defesa alternativas é feita uma análise para escolha da(s) medida(s) mais adequada(s). Uma alternativa de operacionalização, que tem impacto positivo para o requisito não-funcional que se deseja operacionalizar, pode ter impacto negativo em outros requisitos não-funcionais.

A escolha da alternativa de operacionalização mais adequada é norteada pela priorização dos requisitos não-funcionais entre si, considerando-se também probabilidade de uma vulnerabilidade vir a ser explorada, ou seja, do risco que a vulnerabilidade configura vir a se concretizar. Este passo faz parte das atividades de modelagem e análise de requisitos.
❻ Passo iterativo: identificação de outras medidas de ataque – A adição de medidas de defesa pode trazer novas vulnerabilidades ao sistema. É necessário fazer um novo ciclo de identificação de vulnerabilidades, a fim do qual, uma nova análise de medidas de defesa será disparada.
❼ Incorporação das medidas de defesa escolhidas – As medidas de defesa escolhidas, dentre as alternativas estudadas, são incorporadas ao modelo finalizando a modelagem dos requisitos de segurança. Com a incorporação das medidas de defesa, os modelos SR são atualizados disparando em seqüência a atualização dos Cenários, via passo 9 (definição de requisitos gerais), integrando os requisitos de segurança com as soluções escolhidas à definição dos requisitos gerais.

Ao final deste processo, os cenários terão incorporados de forma detalhada os requisitos de segurança tratados e as respectivas soluções adotadas. Estes cenários detalhados podem ser usados na validação dos requisitos (tantos os gerais, quantos os específicos de segurança). A validação dos requisitos [33] pode ser feita através da leitura dos requisitos, expressos na forma de cenários, pelos clientes e demais interessados. Conforme mencionado anteriormente, acreditamos que para clientes e interessados que não estejam bem familiarizados com o *framework* i* seja mais fácil a validação dos requisitos através da leitura dos cenários que pela leitura dos modelos SD, SR e SA.

4
Estudo de Caso

Este capítulo apresenta a aplicação do processo evoluído em um estudo de caso. Inicialmente é apresentada uma breve introdução com a justificativa para o caso escolhido, seguida da descrição do caso. A seção seguinte apresenta os resultados da aplicação passo a passo do processo evoluído, proposto no capítulo anterior.

4.1 Introdução

O objetivo da elaboração de um estudo de caso é validar o processo evoluído, proposto no capítulo anterior. Este estudo de caso trata especificamente de dois sub-tipos de requisitos de segurança da informação: confidencialidade e consistência. Embora disponibilidade também seja um requisito de segurança essencial, confidencialidade e consistência são questões mais centrais em segurança da informação. Outro motivo para focar o estudo de caso em confidencialidade e consistência das informações é que o manuseio da informação pode ser claramente modelado em i*, uma vez que existe em i* uma associação clara entre informação e recurso (entidade informacional), e entre processo de manuseio da informação e tarefa (um modo particular de produzir/manusear um recurso). Confidencialidade e consistência são requisitos que dependem diretamente do manuseio da informação. Para tratar outros requisitos de segurança, como disponibilidade, além da informação outros atores e recursos precisam ser cuidadosamente modelados, como hardware, sistema operacional, infra-estrutura de rede, servidores de aplicação, servidores *web*, sendo estes outros atores e recursos o foco da atenção despendida na definição de requisitos específicos de segurança.

4.2 O Caso

O estudo de caso desenvolvido neste trabalho é definição de requisitos (gerais e específicos de segurança) do *Expert Committee*. O *Expert Committe* foi escolhido para o estudo de caso por ter um escopo com tamanho adequado para o estudo de caso (cinco atores identificados inicialmente) e por possuir informações (tais como: artigo, artigo revisado) para as quais os requisitos de confidencialidade e consistência da informação são perfeitamente aplicáveis.

O *Expert Committee* (EC) é um sistema para suporte ao gerenciamento de submissões e revisões de artigos submetidos a uma conferência ou workshop. Agentes de software foram introduzidos no sistema EC para assistir os pesquisadores e a comissão de organização de um evento em tarefas que fazem parte do processo de revisão e submissão e que podem ser automatizadas. A descrição do problema usado no estudo de caso é a mesma da disciplina Engenharia de Software de Sistemas Multi-Agentes do curso de mestrado, Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro [34]. As atribuições dos envolvidos no *Expert Committee* estão descritas a seguir:

Autor - O autor de um artigo envia o artigo para o evento e espera a resposta da revisão. No processo de submissão do artigo, o autor deve informar título do artigo, abstract, autores e instituições e a(s) área(s) na(s) qual (quais) o artigo se classifica. Caso o artigo seja aceito, o autor envia o camera-ready do artigo, i.e., a versão final do artigo.

Chair - A principal atribuição do *chair* é distribuir os artigos para os revisores e enviar o resultado das revisões para os autores. O *chair* distribui os artigos para os revisores de acordo com a área de interesse dos revisores e com a(s) área(s) na(s) qual (quais) o artigo se encontra. Cada artigo deve ser revisto por no mínimo 3 revisores e cada revisor revê no máximo 3 artigos. Além disso, revisores não podem revisar artigos de autores da mesma instituição à qual ele está associado. Antes de enviar os arquivos dos artigos para os revisores, o chair envia uma proposta de revisão; esta proposta contém uma lista dos títulos dos artigos, os abstracts e as áreas dos mesmos. O revisor avalia a proposta e informa ao chair se aceita ou não revisar tais

artigos. Caso algum revisor não aceite revisar algum artigo, o *chair* entra em contato com o coordenador geral pedindo novos revisores. O coordenador geral envia uma nova lista de revisores para o chair. O *chair* analisa esta lista, seleciona revisores para os artigos que não possuem 3 revisores e envia a proposta para cada revisor selecionado. Para simplificar o problema assumimos que nesta segunda etapa de proposta de revisão nenhum revisor irá rejeitar a proposta. Uma vez que todos os artigos estejam alocados para revisores, o *chair* envia os arquivos contendo os artigos (assume-se um arquivo para cada artigo). Após a revisão, os revisores a enviam ao *chair*. Com base nesses resultados, o *chair* verifica se existe algum conflito. Um conflito ocorre quando um artigo é fortemente aprovado por um revisor e é fortemente reprovado por um outro revisor. Para os artigos com revisões em conflito, o *chair* envia artigos e revisões para os membros do comitê de programa. Os membros do comitê votam nos artigos que devem ser aceitos e enviam a votação para o chair; por exemplo, um membro indica se aceita ou rejeita o artigo. O *chair* analisa os votos e gera o resultado final. Em seguida, o *chair* envia as revisões e o resultado final de cada artigo aos autores. No caso dos artigos aceitos, o *chair* recebe o câmera-ready destes artigos.

Revisor - O revisor tem por atribuição rever 3 dos artigos submetidos a um evento, e que lhe foram atribuídos pelo *chair*. Quando o revisor recebe a lista de artigos que deve rever, ele verifica se pode ou não aceitar a proposta de acordo com os compromissos em sua agenda, com o deadline para envio de revisões e com suas áreas de interesse. Os resultados desta verificação são comunicados pelo revisor ao *chair*. Após concluir a revisão de cada artigo, o revisor envia a revisão para o *chair* juntamente com o seu parecer final sobre o artigo. O parecer final pode ser: fortemente aprovado, aprovado com restrição, fracamente rejeitado e fortemente rejeitado.

Coordenador Geral - O objetivo do coordenador geral no sistema é fornecer revisores para o *chair* de um evento. Quando o *chair* pede novos revisores, ele informa o número de revisores que deseja e a área de interesse dos novos revisores. O coordenador geral consulta sua base de dados e fornece no mínimo 1,5 vezes o número de revisores pedidos pelo *chair*.

Membro do Comitê de Programa - Os membros do comitê de programa avaliam os conflitos existentes nas revisões dos artigos. Eles recebem do *chair* os artigos onde ocorreram conflitos e suas respectivas revisões. Com base nas revisões, cada um vota pela aceitação ou rejeição do artigo. Esta informação é enviada ao *chair* do evento.

4.3 O Estudo – Aplicação do processo

Para este estudo de caso a elicitação dos requisitos foi baseada principalmente na descrição do *Expert Committe* [34], e na elicitação e parte da modelagem apresentada em [4]. Em virtude disto, a técnica usada na elicitação foi basicamente a leitura de documentos, auxiliada pelo uso de senso comum do engenheiro de requisitos e por alguma liberdade para inventar, quando necessário.

O resultado da aplicação do processo, passos da Figura 3.3, ao caso do *Expert Committee* é apresentado, passo a passo, em seguida:

Passo 1 (Definição de requisitos gerais) – Identificação do Atores

Com base na descrição do *Expert Committee* foram identificados os atores apresentados na tabela 4.1, a seguir:

Ator	Fonte	Iteração
Chair	Descrição do problema	Primeira
Autor	Descrição do problema	Primeira
Revisor	Descrição do problema	Primeira
Membro do Comitê de Programa	Descrição do problema	Primeira
Coordenador Geral	Descrição do problema	Primeira

Tabela 4-1 - Identificação de Atores – versão inicial

Passo 1 (Definição de requisitos específicos de segurança) – Identificação de Atacantes

Para cada ator identificado no passo anterior é identificado um atacante correspondente. Além destes atacantes, foi acrescentado também um atacante

externo (à conferência). A tabela 4.2 apresenta os atacantes identificados inicialmente:

Atacante	Fonte	Iteração
Atacante do Chair	Identificação de atores	Primeira
Atacante do Autor	Identificação de atores	Primeira
Atacante do Revisor	Identificação de atores	Primeira
Atacante do Membro do Comitê	Identificação de atores	Primeira
Atacante do Coordenador Geral	Identificação de atores	Primeira
Atacante Externo	-	Primeira

Tabela 4-2 - Identificação de Atacantes – versão inicial

Passo 2 (Definição de requisitos gerais) – Identificação das dependências entre atores

Neste passo é elaborado o modelo SD com base na descrição do problema e nos atores identificados no passo 1 – Identificação de Atores e no passo 2 – Identificação de dependências entre atores. Como mencionado anteriormente, o objetivo deste modelo é representar as dependências estratégicas entre os atores.

As relações de dependência estratégica (entre atores), apresentadas no modelo SD da Figura 4.1 (reusado da modelagem apresentada em [4]) estão descritas a seguir.

Chair x Coordinator:

O *Chair* depende do Coordenador (*Coordinator*) para satisfazer seus desejos de que sejam indicados os revisores (representado pela meta '*Reviewers Be Indicated*') e membros do comitê de programa (representado pela meta '*Committee Member Be Indicated*'), e o desejo de que as indicações sejam boas (com qualidade), o que é representado pela meta flexível '*Quality [good indication]*'. O Coordenador depende que o *Chair* lhe disponibilize as datas do evento, representada no modelo pelo recurso '*Event Dates*'.

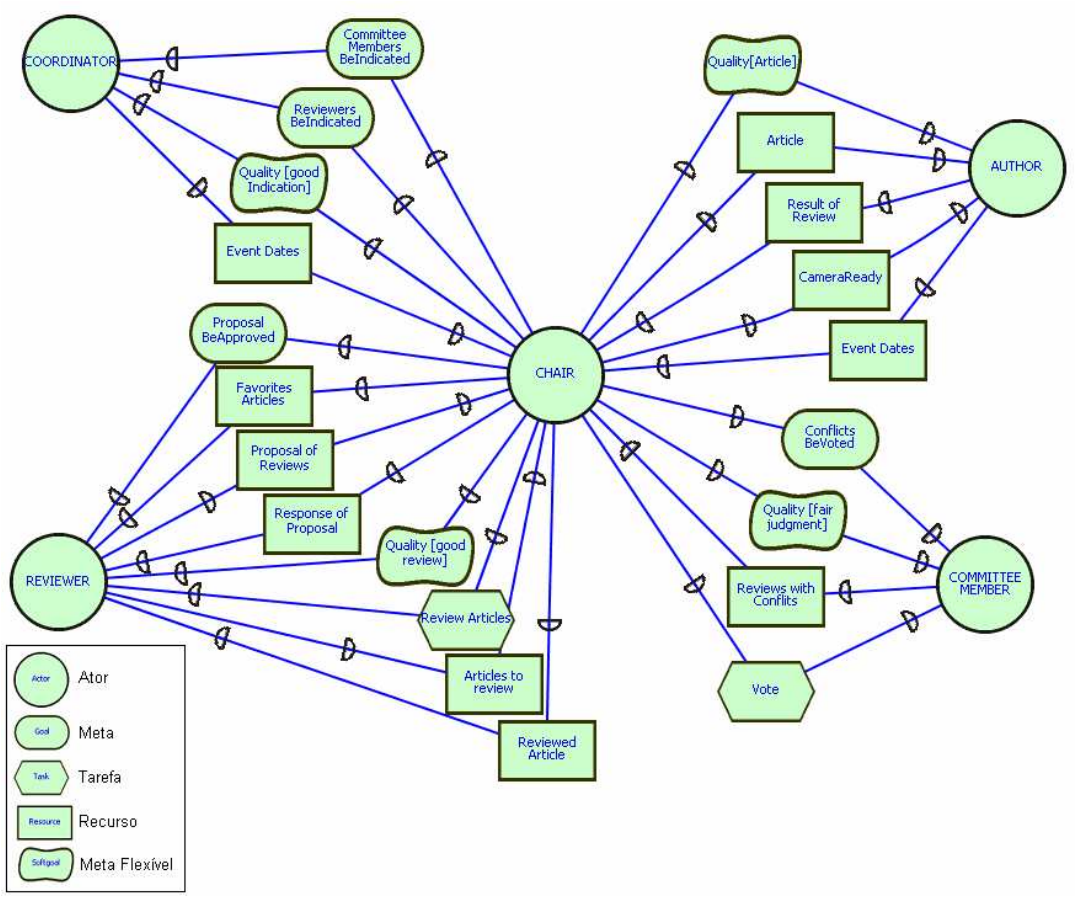

Figura 4.1 - Modelo SD do *Expert Committee*, reusado de [4]

Chair x Author:

O *Chair* depende do Autor (*Author*) para que este submeta artigos à conferência, e caso o artigo submetido seja aceito pela conferência, o *Chair* depende também do Autor para que este envie a versão final. O artigo submetido inicialmente e a versão final estão representados no modelo SD pelos recursos '*Article*' e '*Camera Ready*', respectivamente. O *Chair* depende também do Autor para que os artigos tenham qualidade, o que é representado no modelo pela meta flexível '*Quality [Article]*'. O Autor depende do *Chair* para saber as datas e prazos do evento e para obter uma resposta para a submissão do seu artigo. As datas e prazos são representados pelo recurso '*Event Dates*' e a resposta pelo recurso '*Result of Review*'.

Chair x Reviewer:

O Revisor (*Reviewer*) depende que o *Chair* lhe envie propostas de revisão (representadas pelo recurso '*Proposal of Reviews*') ao passo que o *Chair* depende do Revisor para saber seus artigos favoritos (representado pelo recurso '*Favorites Articles*') e para obter uma resposta (representada pelo recurso '*Response of Proposal*') para uma proposta de revisão. O *Chair* deseja também que as propostas de revisão que ele envia ao Revisor sejam aprovadas, sendo este desejo representado pela meta '*Proposal Be Approved*' - que o *Chair* depende do Revisor para satisfazer.

O *Chair* depende que o Revisor faça as revisões do artigo. Esta dependência foi modela através da tarefa '*Review Articles*', por haver alguns algumas considerações que o Revisor deve seguir ao efetuar uma revisão. O *Chair* depende do Revisor também para que as revisões sejam de boa qualidade, o que está representado no modelo pela meta flexível '*Quality [good review]*'. O Revisor depende que o *Chair* disponibilize os artigos para revisão (representados pelo recurso '*Articles to review*'), enquanto o *Chair depende* que o Revisor envie os artigos revisados (representados pelo recurso '*Reviewed Article*').

Chair x Committee Member

O *Chair* deseja que os conflitos em revisões sejam solucionados, desejo representado no modelo pela meta '*Conflict Be Voted*', e para isto depende do Membro de Comitê de Programa (*Committee Member*). O *Chair* necessita que o Membro de Comitê de Programa vote, nos casos de conflito nas revisões. O ato de votar está representado pela tarefa '*Vote*', que o *Chair* depende que o Membro do Comitê de Programa desempenhe. A opção de representar o voto através de uma tarefa e não de um recurso se deve ao fato de haver alguns critérios a serem seguidos para a votação. Além disto, o *Chair* necessita que a votação seja justa, o que está representado no modelo pela meta flexível '*Quality [Fair judgment]*'.

Passo 3 (Definição de requisitos gerais) – Definição de *SDsituations*

Através da observação do modelo SD (Figura 4.1) e do reuso da elicitação e parte da modelagem feitas em [4], foram identificadas as seguintes situações de dependências (*SDsituations*):

- Indicação dos Membros do Comitê de Programa (‘*Committee Indication*’)
- Indicação dos Revisores (‘*Reviewers Indication*’)
- Submissão de Artigo (‘*Article Submission*’)
- Aceitação de Proposta de Revisão (‘*Proposal Acceptance*’)
- Revisão de Artigos (‘*Article Reviews*’)
- Resolução de Conflitos (‘*Conflicts Solution*’)
- Recepção das Versões Finais (‘*Camera Ready Reception*’).

Estas *SDsituations* possuem interdependências temporais, representadas no diagrama de *SDsituations* [4] (Figura 4.2). As primeiras situações a ocorrer são: ‘Indicação dos Membros do Comitê de Programa’ e ‘Indicação dos Revisores’, que ocorrem paralelamente. Após a finalização destes ciclos e decorrido um intervalo de tempo (representado no diagrama por T1) inicia-se o ciclo da situação ‘Submissão de Artigo’ seguido, após seu encerramento, pelo ciclo de ‘Aceitação de Proposta de Revisão’. Uma vez finalizado o ciclo de ‘Aceitação de Proposta de Revisão’ inicia-se o ciclo de ‘Revisão de Artigo’, e após um intervalo de tempo (representado no diagrama como T2) inicia-se também o ciclo de ‘Solução de Conflitos’ que corre em paralelo ao ciclo de ‘Revisão de Artigo’ iniciado antes. Após o encerramento simultâneo destes dois ciclos, observado o intervalo de tempo T3, inicia-se o ciclo da última situação: ‘Recepção das Versões Finais’. As *SDsituations* estão descritas detalhadamente, em *frames* [4], a seguir:

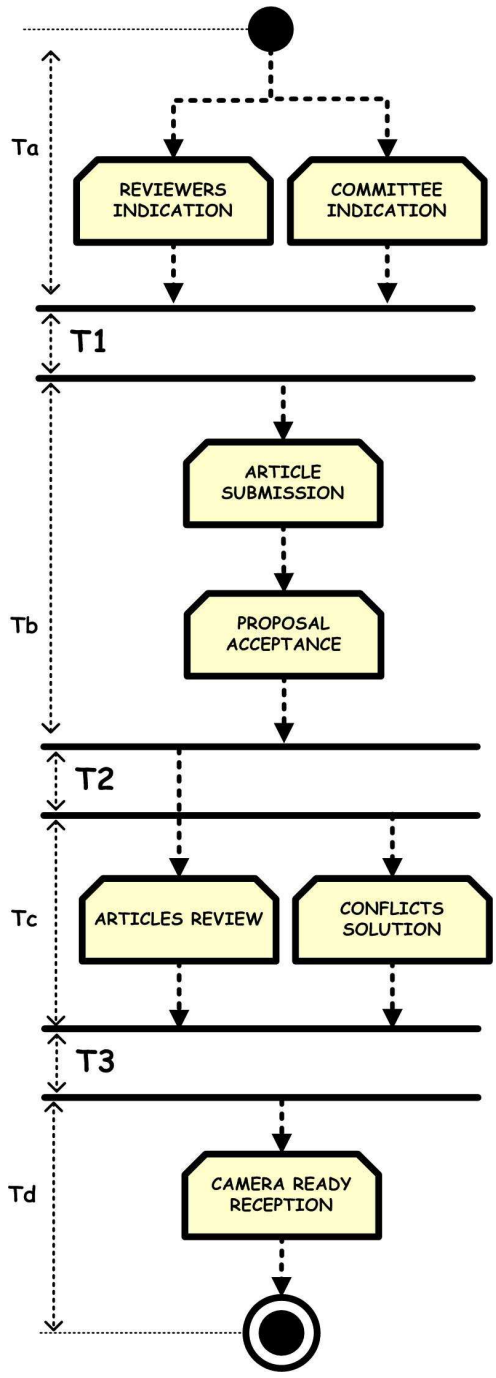

Figura 4.2 - Diagrama de *SDSituations* do *Expert Committee,* adaptado de [4]

SDsituation Definition				
SDsituation:	**COMMITTEE MEMBERS INDICATION**			
Goal:	Committee Members be indicated.			
Definition:	It represents a situation when the Coordinator indicates Committee Members to the conference.			
ACTOR (depender)	ELEMENTS OF DEPENDENCY	TYPE	DEGREE	ACTOR (dependee)
	REASEARCHERS INVITATION	DEPENDENCY	critical	
Chair	Conference Members Be Indicated	goal	critical	Coordinator
Chair	Quality [Good Indication]	softgoal	critical	Coordinator
Coordinator	Event Dates	resource	critical	Chair

SDsituation Definition				
SDsituation:	**REVIEWERS INDICATION**			
Goal:	Reviewers be indicated.			
Definition:	It represents a situation when the Coordinator indicates Reviewers to the conference.			
ACTOR (depender)	ELEMENTS OF DEPENDENCY	TYPE	DEGREE	ACTOR (dependee)
	REASEARCHERS INVITATION	DEPENDENCY	critical	
Chair	ReviewersBeIndicated	goal	critical	Coordinator
Chair	Quality [Good Indication]	softgoal	critical	Coordinator
Coordinator	Event Dates	resource	critical	Chair

SDsituation Definition				
SDsituation:	**ARTICLE SUBMISSION**			
Goal:	Article be submitted			
Definition:	It represents a situation when the Author submits an article to the conference.			
ACTOR (depender)	ELEMENTS OF DEPENDENCY	TYPE	DEGREE	ACTOR (dependee)
Chair	Quality [Article]	softgoal	critical	Author
Author	Event Dates	resource	critical	Chair
Chair	Article	resource	critical	Author

SDsituation Definition				
SDsituation:	**PROPOSAL ACCEPTANCE**			
Goal:	Proposal be accepted			
Definition:	It represents a situation when the Chair asks reviewers to accept a proposal of review.			
ACTOR (depender)	ELEMENTS OF DEPENDENCY	TYPE	DEGREE	ACTOR (dependee)
	REVIEWERS INDICATION	DEPENDENCY	critical	
	ARTICLE SUBMISSION	DEPENDENCY	critical	
Chair	ProposalBeApproved	goal	critical	Reviewer
Chair	Favorites Articles	resource	critical	Reviewer
Reviewer	Proposal of Review	resource	critical	Chair
Chair	Response of Proposal	resource	critical	Reviewer

SDsituation Definition				
SDsituation:	**ARTICLES REVIEW**			
Goal:	ArticlesBeReviewed			
Definition:	It represents a situation when the Reviewer reviews an article (the proposal of review has been previously accepted)			
ACTOR (depender)	ELEMENTS OF DEPENDENCY	TYPE	DEGREE	ACTOR (dependee)
	PROPOSAL ACCEPTANCE	DEPENDENCY	critical	
Chair	Review Article	task	critical	Reviewer
Chair	Quality [Good Review]	softgoal		Reviewer
Reviewer	Article to Review	resource	critical	Chair
Chair	Reviewed Article	resource	critical	Reviewer

SDsituation Definition				
SDsituation:	**CONFLICTS SOLUTION**			
Goal:	Conflicts be solved			
Definition:	It represents a situation when the Committee votes in the cases of reviews with conflict (positives and negatives reviews for the same article)			
ACTOR (depender)	ELEMENTS OF DEPENDENCY	TYPE	DEGREE	ACTOR (dependee)
	ARTICLES REVIEW	DEPENDENCY	critical	
Chair	ConflictsBeVoted	goal	critical	Committee Member
Chair	Quality [Fair judgment]	softgoal		Committee Member
Chair	Vote	task		Committee Member
Committee Member	Reviews with conflict	resource	critical	Chair

SDsituation Definition				
SDsituation:	**CAMERA READY RECEPTION**			
Goal:	Camera Ready be received			
Definition:	It represents a situation when the Author sents the Camera Ready (of an article previously accepted).			
ACTOR (depender)	ELEMENTS OF DEPENDENCY	TYPE	DEGREE	ACTOR (dependee)
	ARTICLES REVIEW	DEPENDENCY	critical	
	CONFLICTS SOLUTION	DEPENDENCY		
Chair	Quality [Article]	softgoal	critical	Author
Author	Event Dates	resource	critical	Chair
Author	Result of Review	resource	critical	Chair
Chair	Camera Ready	resource	critical	Author

Passo 4 – iterativo (Definição de requisitos gerais) - Identificação de novos atores

Examinando-se as *SDsituations* 'Indicação dos Revisores' e 'Indicação dos Membros do Comitê de Programa' foi identificada a necessidade de mais um ator: Pesquisador. A necessidade de representar este novo ator surgiu em virtude das dependências estratégicas que Coordenador Geral mantém com o *Chair*. Com isto os passos 1 (Identificação de Atores), 2 (Identificação de dependências entre atores) foram revistos. A revisão do passo 1 gerou uma nova versão para a tabela de identificação de atores, e conseqüentemente para a tabela de identificação de atacantes, apresentadas nas tabelas 4.3 e 4.4 a seguir:

Ator	Fonte	Iteração
Chair	Descrição do problema	Primeira
Autor	Descrição do problema	Primeira
Revisor	Descrição do problema	Primeira
Membro do Comitê de Programa	Descrição do problema	Primeira
Coordenador Geral	Descrição do problema	Primeira
Pesquisador	Modelo SD	Segunda

Tabela 4-3 - Identificação de Atores – versão revisada

Atacante	Fonte	Iteração
Atacante do Chair	Identificação de atores	Primeira
Atacante do Autor	Identificação de atores	Primeira
Atacante do Revisor	Identificação de atores	Primeira
Atacante do Membro do Comitê	Identificação de atores	Primeira
Atacante do Coordenador Geral	Identificação de atores	Primeira
Atacante Externo		Primeira
Atacante do Pesquisador	Modelo SD	Segunda

Tabela 4-4 - Identificação de Atacantes – versão revisada

Na revisão do Passo 2 (Identificação de dependências entre atores), o Pesquisador e seus relacionamentos de dependência estratégica foram acrescentados ao modelo SD, gerando uma nova versão apresentada na Figura 4.3.

Neste novo modelo SD, o Coordenador Geral (*Coordinator*) depende do Pesquisador (*Researcher*) para que um convite para participação na conferência seja aceito. No modelo SD o convite (*Invitation*), feito pelo Coordenador, e a resposta (*Response of Invitation*) dada pelo Pesquisador são representados por recursos, enquanto o desejo do Coordenador de 'ter o convite aceito' é representado pela meta '*Invitation Be Accepted*', que o Coordenador depende do Pesquisador para satisfazer.

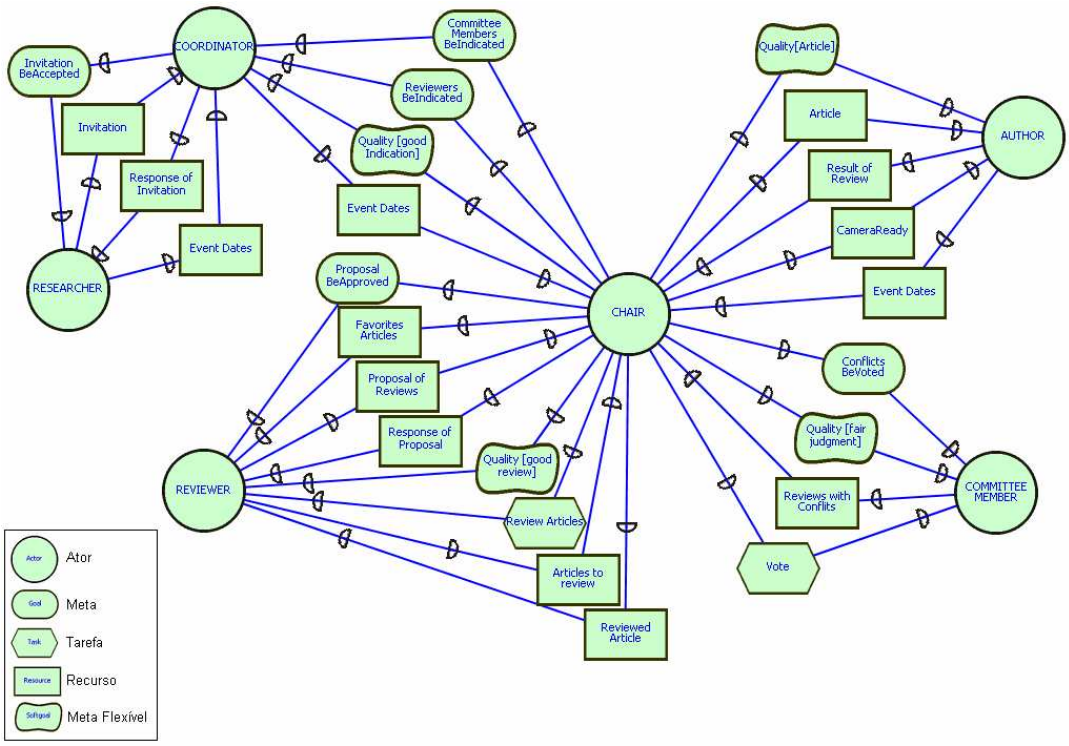

Figura 4.3 - Modelo SD do *Expert Committee* – versão revisada

Na revisão do passo 3 (Definição de *SDsituations*), foi identificada mais uma situação de dependência: Convite a pesquisadores ('Researchers Invitation'). Esta situação é a primeira situação a ocorrer, antes das situações: 'Indicação dos Membros do Comitê de Programa' e 'Indicação dos Revisores', que ocorrem paralelamente. Com a inclusão desta nova *SDsituation*, foi gerada uma nova versão para o diagrama de *SDsituations*, apresentado na Figura 4.4.

A nova *SDsituation* identificada está descrita a seguir.

SDsituation Definition				
SDsituation:	**RESEARCHERS INVITATION**			
Goal:	Invitation be accepted			
Definition:	It represents a situation when the Chair asks researchers to accept an invitation to join the conference.			
ACTOR (depender)	ELEMENTS OF DEPENDENCY	TYPE	DEGREE	ACTOR (dependee)
Coordinator	InvitationBeAccepted	goal	critical	Researcher
Researcher	Invitation	resource	critical	Coordinator
Coordinator	Response of Invitation	resource	critical	Researcher

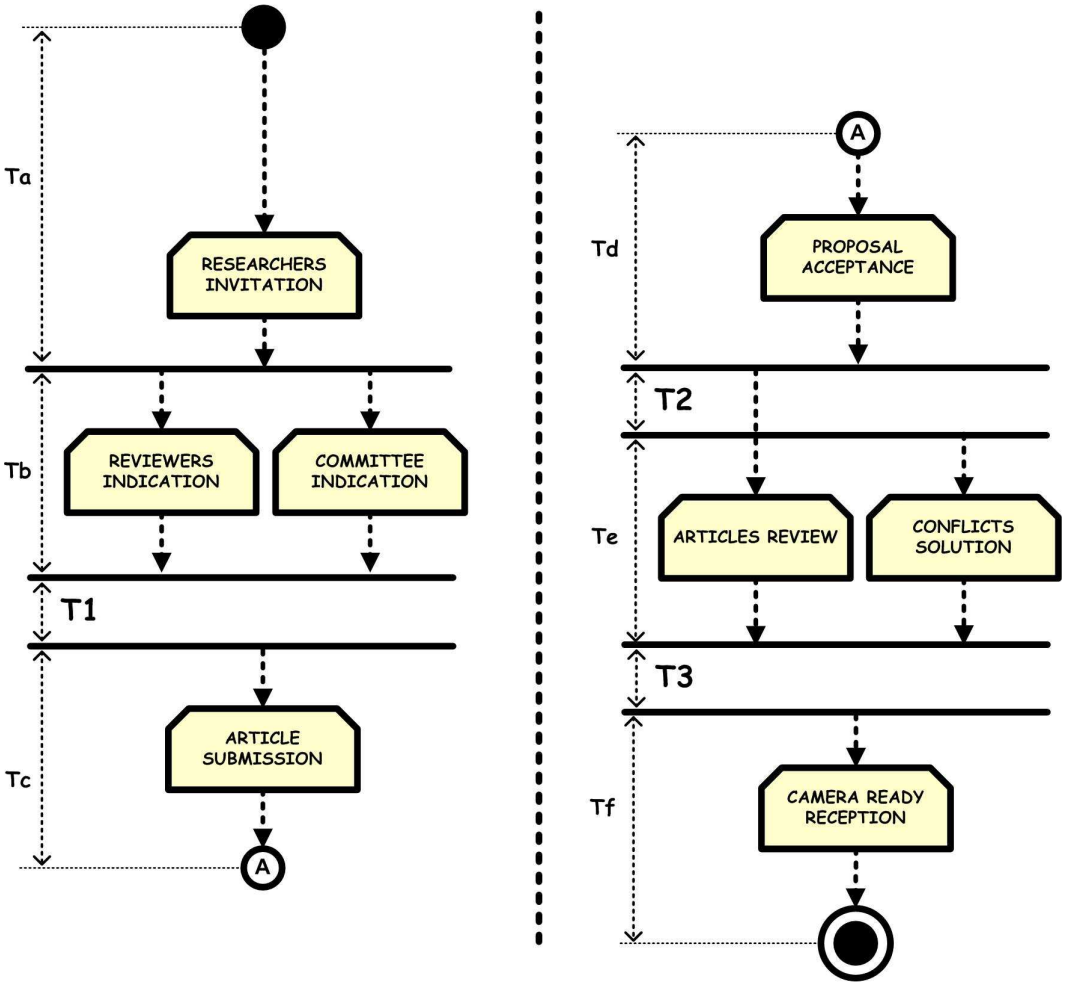

Figura 4.4 - Diagrama de *SDSituations* do *Expert Committee,* adaptado de [17]

Passo 2 (Definição de requisitos específicos de segurança) – Identificação de vulnerabilidades

Este passo consiste em identificar os pontos de vulnerabilidade nas relações de dependência entre os atores. Como o foco deste estudo de caso é confidencialidade e consistência das informações, a investigação das vulnerabilidades do sistema inicia-se pela identificação das informações (recursos em i*) que fazem parte dos relacionamentos de dependência entre os atores. Além dos recursos em si, tarefas diretamente associadas com a produção de alguma informação valiosa também devem ser analisadas. Estes recursos e tarefas são identificados a partir do modelo SD e das *SDsituations* produzidos anteriormente, da seguinte forma:

- Identificar os elementos de dependência (*dependum*) de cada *SDsituation* do tipo recurso ou tarefa;

- Para cada elemento identificado, inferir se é (uma tarefa que produz) um recurso sensível à segurança (recursos sensíveis à segurança devem ser protegidos, configurando pontos de vulnerabilidade). *Um recurso sensível à segurança é um recurso que possui ao menos uma das seguintes características:*

  - *seu conteúdo não pode ser perdido ou adulterado (consistência);*

  - *o acesso (ao recurso) deve ser restrito (confidencialidade).*

  Esta análise pode ser feita usando-se como base a classificação das informações em relação à segurança, caso a organização tenha uma política de segurança da informação implementada.

A Tabela 4.5 apresenta uma lista dos elementos de dependência, oriundos do modelo SD, candidatos a serem identificados como ponto de vulnerabilidade:

Elementos de dependência	**Tipo**	***SDsituations***	**Deve ser protegido?**
Invitation	Resource	RESEARCHERS INVITATION	Não
Response of Invitation	Resource	RESEARCHERS INVITATION	Não
Event Dates	Resource	RESEARCHERS INVITATION, COMMITTEE M. INDICATION, REVIEWERS INDICATION, ARTICLE SUBMISSION, CAMERA READY RECEPTION	Não
Article	**Resource**	**ARTICLE SUBMISSION,**	**Sim**

		ARTICLES REVIEW,	
Favorites Articles	Resource	PROPOSAL ACCEPTANCE	Não
Proposal of Review	Resource	PROPOSAL ACCEPTANCE	Não
Response of Proposal	Resource	PROPOSAL ACCEPTANCE	Não
Review Article	**Task**	**ARTICLES REVIEW**	**Sim**
Reviewed Article	**Resource**	**ARTICLES REVIEW, CAMERA READY RECEPTION, CONFLICTS SOLUTION**	**Sim**
Vote	**Task**	**CONFLICTS SOLUTION**	**Sim**
Result of Review	**Resource**	**ARTICLES REVIEW, CONFLICTS SOLUTION, CAMERA READY RECEPTION,**	**Sim**
Camera Ready	**Resource**	**CAMERA READY RECEPTION**	**Sim**

Tabela 4-5 **-** Identificação de vulnerabilidades

Na análise feita neste estudo de caso, recursos como o convite (*Invitation*) enviado pelo *Chair* aos pesquisadores ou datas programadas para o evento (*Event Dates*) não foram considerados como valiosos o suficiente para merecerem proteção. Já a versão final de um artigo (*Câmera Ready*), embora não seja uma informação sensível a confidencialidade (não é necessário restringir o acesso a ela, uma vez que o destino da versão final é ser publicada), é uma informação sensível em relação à consistência da informação (seu conteúdo não pode ser perdido ou adulterado). Os elementos de dependência identificados como sensíveis na Tabela 4.5, que constituem pontos de vulnerabilidade ao sistema, são:

⇒ Recursos: Artigo (*Article*), Artigo Revisado (*Reviewed Article*), Resultado da Revisão (*Result of Review*) e Versão final do artigo (*Camera Ready*).

⇒ Tarefas: Revisar Artigo (*Review Article*) e Votar (*Vote*).

Passo 5 (Definição de requisitos gerais) – Definição de Cenários

Conforme mencionado anteriormente, o usual é descrever um cenário para cada *SDsituation* [4]. Os cenários [5] (versões iniciais) correspondentes as *SDsituations*, identificadas no passo 4 – Definição de *SDsituations*, estão descritos a seguir:

Scenario definition	
Title:	**Researchers Invitation** (incomplete)
Goal:	Invitation be accepted.
Context:	

Geographical Location:	WEB
Temporal Location:	Before the beginning the article submission phase.
Constraint:	
Precondition:	
Resources:	Computer, Internet, event dates
Constraint:	
Actors:	Coordinator, Researcher
Episodes:	Coordinator prepares the invitations.
	Coordinator sends the invitations to Researchers.
	Researcher sends the response of invitation to Coordinator
	Coordinator receives the responses and prepare the reviewers list and the committee members list.
Constraint:	NFR: Security, User-friendliness
Exceptions:	

Scenario definition	
Title:	**Reviewers Indication** (incomplete)
Goal:	Reviewers be indicated.
Context:	
Geographical Location:	WEB
Temporal Location:	Until the reviews deadline.
Constraint:	
Precondition:	Invitation Acceptance
Resources:	Computer, Internet
Constraint:	
Actors:	Chair, Coordinator
Episodes:	Chair ask Coordinator for Reviewers indication (specifying the areas)
	Chair informs the event dates to Coordinator.
	Coordinator prepare the Reviewers indication (according to the specified areas)
	Coordinator send the Reviewers indication to Chair
	Chair receives the Reviewers indication.
Constraint:	NFR: Security, User-friendliness
Exceptions:	

Scenario definition	
Title:	**Article Submission** (incomplete)
Goal:	Article be submitted.
Context:	
Geographical Location:	WEB
Temporal Location:	Before the submission deadline.
Constraint:	

	Precondition:	
Resources:	Computer, Internet, Article	
	Constraint:	
Actors:	Chair, Author	
Episodes:	Author prepare the article.	
	Author sends the article to Chair.	
	Chair receives the article	
	Constraint:	NFR: Security, User-friendliness
Exceptions:		

Scenario definition		
Title:	**Committee Members Indication** (incomplete)	
Goal:	Committee Members be indicated	
Context:		
	Geographical Location:	WEB
	Temporal Location:	Until the reviews deadline.
	Constraint:	
	Precondition:	Invitation Acceptance
Resources:	Computer, Internet	
	Constraint:	
Actors:	Chair, Coordinator	
Episodes:	Chair ask Coordinator for Committee Members indication (specifying the areas)	
	Chair informs the event dates to Coordinator.	
	Coordinator prepare the Committee Members indication(according to the specified areas)	
	Coordinator send the Committee Members indication to Chair	
	Chair receives the Committe Members indication.	
	Constraint:	NFR: Security, User-friendliness
Exceptions:		

Scenario definition		
Title:	**Proposal Acceptance** (incomplete)	
Goal:	Proposal be accepted	
Context:		
	Geographical Location:	WEB
	Temporal Location:	Right after the submission deadline.
	Constraint:	
	Precondition:	Reviewers list and articles list have been prepared
Resources:	Computer, Internet, reviewers list, articles list	
	Constraint:	
Actors:	Chair, Reviewers	
Episodes:	Chair prepare proposals	

	Chair selects reviewers within the same area of the article.
	Chair separates out reviewers of the same institution of the author.
	Chair makes proposals to reviewers.
	Chair sends proposals to each *reviewer giving the acceptance deadline.
	Chair receives answered proposals form reviewers
	Chair verifies answered proposals.
Constraint:	NFR: Security, User-friendliness
	* Each reviewer can not receive more than 3 articles.
Exceptions:	If there is at least one article without 3 reviewers:
	(scenario: “Reviewers Indication”)

Scenario definition		
Title:	**Articles Review** (incomplete)	
Goal:	ArticlesBeReviewed	
Context:		
Geographical Location:	WEB	
Temporal Location:	Before deadline to send reviews.	
Constraint:		
Precondition:	Proposal Acceptance.	
Resources:	Computer, Internet, Articles to review	
Constraint:		
Actors:	Chair, Reviewers	
Episodes:	Chair selects an article to sends to review	
	Chair selects reviewers that have accepted the proposal of review for the selected article;	
	Chair sends the article to each reviewer selected (in previous step)	
	Reviewer receives the article sent by Chair	
	Reviewer makes the article review	
	Reviewer sends reviewed article to Chair	
	Chair receives the reviewed articles	
	Chair checks if there is a conflict (incompatible evaluations) in the reviews of each article	
Constraint:	NFR: Security, User-friendliness	
Exceptions:	If an article has a conflict:	
	(scenario: Conflict Resolution)	

Scenario definition		
Title:	**Conflict Solution** (incomplete)	
Goal:	Conflict be solved.	
Context:		
Geographical Location:	WEB	
Temporal Location:	After articles have been reviewed.	
Constraint:		
Precondition:	Review Article	
Resources:	Computer, Internet, Articles, Reviews	
Constraint:		
Actors:	Chair, Committee Members	
Episodes:	Chair selects an article with a conflict (incompatible evaluations)	
	Chair selects Committee Members in the same area of the article.	
	Chair separates out reviewers of the same institution of the Author.	
	Chair sends reviews with conflict to each selected Committee Member giving the deadline to vote	
	Committee Members receives the reviews with conflict from Chair	
	Committee Members votes (to solve the conflict)	

	Chair receives the votes	
	Constraint:	NFR: Security, User-friendliness
Exceptions:		

Scenario definition		
Title:	**Camera Ready Reception** (incomplete)	
Goal:	Camera Ready be sent.	
Context:		
Geographical Location:		WEB
Temporal Location:		Before the camera ready sending deadline.
Constraint:		
Precondition:		Article be accepted
Resources:	Computer, Internet, camera ready	
	Constraint:	
Actors:	Author, Chair	
Episodes:	Chair sends the result of review to the Author.	
	Author receives the result of review	
	Author prepares the camera ready.	
	Author sends the camera ready to Chair.	
	Chair receives the camera ready	
	Constraint:	NFR: Security, User-friendliness
Exceptions:		

Passo 6 (Definição de requisitos gerais) – Refinamento de Atores

Os atores (legítimos) e atacantes identificados previamente (tabelas 4.3 e 4.4 respectivamente) são refinados em papéis, posições e agentes nos modelos SA. Este refinamento é feito segundo as seguintes heurísticas:

- atores que possuam características individuais relevantes para o contexto da aplicação e que não possam ser transferidas para outros atores devem ser desdobrados em agentes;

- para os demais atores no modelo SD, são criados papéis que representem a participação do ator nas *SDSituation*. Para as *SDSituations* complementares (com dependência lógica), pode ser criado apenas um papel.

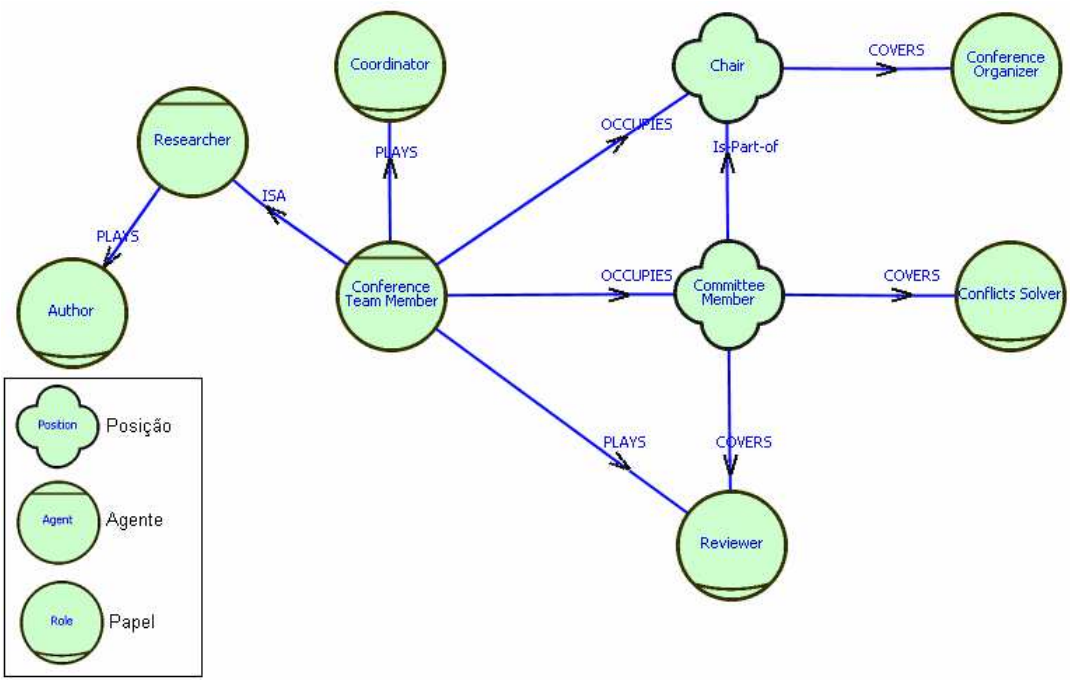

Figura 4.5 - Modelo SA – Atores legítimos

No modelo SA dos atores legítimos, Figura 4.5, o ator Pesquisador (*Researcher*) foi refinado para o agente *Reseacher*, pelo fato de ter características individuais como área de interesse, artigos publicados, instituição, que são relevantes no contexto do *Expert Committe*, e que por serem individuais não poderiam ser transferidas a outro ator. Há no *Expert Committee* um tipo específico de Pesquisador, que desempenha atividades dentro da conferência A este tipo específico de Pesquisador, chamamos de Membro da Equipe da Conferência (*Conference Team Member*). Como um Membro da Equipe da Conferência também é pesquisador, foi criado um agente *Conference Team Member* que tem um relacionamento de "É um" para o agente *Researcher*.

Os atores Autor (*Author*), Coordenador (*Coordinator*) e Revisor (*Reviewer*) foram refinados em papéis, por possuirem características que podem ser transferidas a outros atores no domínio do *Expert Committe* e pelo fato das situações de dependência nas quais estes atores estão envolvidos nos modelos SD poderem ser atribuídas a um único papel em cada caso. O papel *Author* pode ser desempenhado por qualquer Pesquisador, enquanto os papéis *Coordinator e Reviewer* podem ser desempenhados por Membros da Equipe da Conferência (agentes do tipo *Conference Team Member*). O ator Membro do Comitê de

Programa (*Committee Member*) participa da *SDsituation* '*CONFLICTS SOLUTION*' e pode também revisar artigos, participando das mesmas *SDsituations* que o Revisor: '*PROPOSAL ACCEPTANCE*' e '*ARTICLES REVIEW*'. No modelo SA há um papel específico para a *SDsituation* '*CONFLICTS SOLUTION*' chamado 'Resolvedor de Conflitos' (*Conflicts Solver)*. A posição Membro do Comitê (*Committee Member*) apresentada no modelo SA refina o ator Membro do Comitê de Programa identificado anteriormente, cobrindo os papéis de Revisor e de Resolvedor de Conflitos. Esta posição é parte de uma posição maior: a posição de *Chair*. A posição de *Chair* cobre, além dos papéis de Revisor e Resolvedor de Conflitos cobertos pela posição de Membro do Comitê, o papel de Organizador da Conferência (*Conference Organizer*). O papel de Organizador da Conferência representa todas as atribuições que a posição de *Chair* tem a mais que a posição de Membro do Comitê, ou seja, todas as demais situações em que o ator *Chair* está envolvido. Finalmente, as posições de *Chair* e Membro do Comitê podem ser ocupadas por agentes do tipo Membro do Corpo da Conferência. Não há em i* a noção de cardinalidade para os relacionamentos de ocupar (nem desempenhar ou cobrir). Assim, não há como representar no modelo SA que apenas um agente do tipo Membro do Corpo da Conferência pode ocupar a posição de *Chair*.

No modelo SA de atacantes apresentado na Figura 4.6, os atacantes identificados anteriormente (tabela 4.4) foram refinados nos agentes *Chair Attacker, Committee Member Attacker, Author Attacker, Reviewer Attacker, Researcher Attacker, Coordinator Attacker e External Attacker*. Com o objetivo de generalizar algumas características comuns a estes agentes foi criado no SA o agente *Attacker*. A exceção do agente *External Attacker*, todos os demais são atacantes 'internos' a conferência, uma vez que foram concebidos à semelhança dos atores legítimos da conferência. O agente *Internal Attacker* generaliza todos estes atacantes internos. Uma diferença relevante entre estes dois tipos de atacantes é que um agente do tipo *Internal Attacker* constitui ameaça ao sistema por tentar desempenhar os papéis ou ocupar as posições, definidas no modelo SA dos atores legítimos (Figura 4.5), ilicitamente. Uma vez desempenhando tais papéis ou ocupando tais posições, disporia de todas as prerrogativas de um ator legítimo. Já o agente *External Attacker* generaliza agentes que possam empreender ataques ao sistema independentemente de papéis ou posições. Tanto *External Attacker* quanto

Internal Attacker especializam o agente mais genérico *Attacker*, via um relacionamento do tipo “É um”.

.

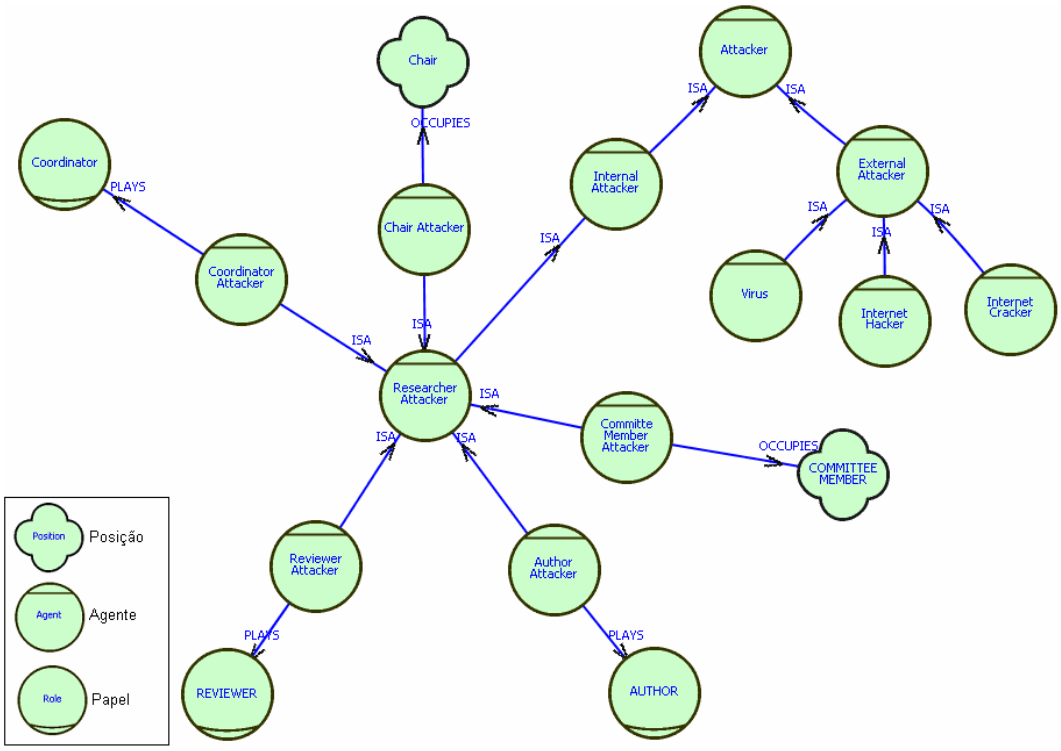

Figura 4.6 - Modelo SA – Atacantes

Passo 7 – iterativo (Definição de requisitos gerais) – Refinamento de modelos SD

Neste passo, os atores existentes no modelo SD elaborado anteriormente (Figura 4.3) são substituídos por agentes, papéis e posições resultantes do refinamento elaborado no passo anterior. O modelo SD refinado é apresentado na Figura 4.7 adiante.

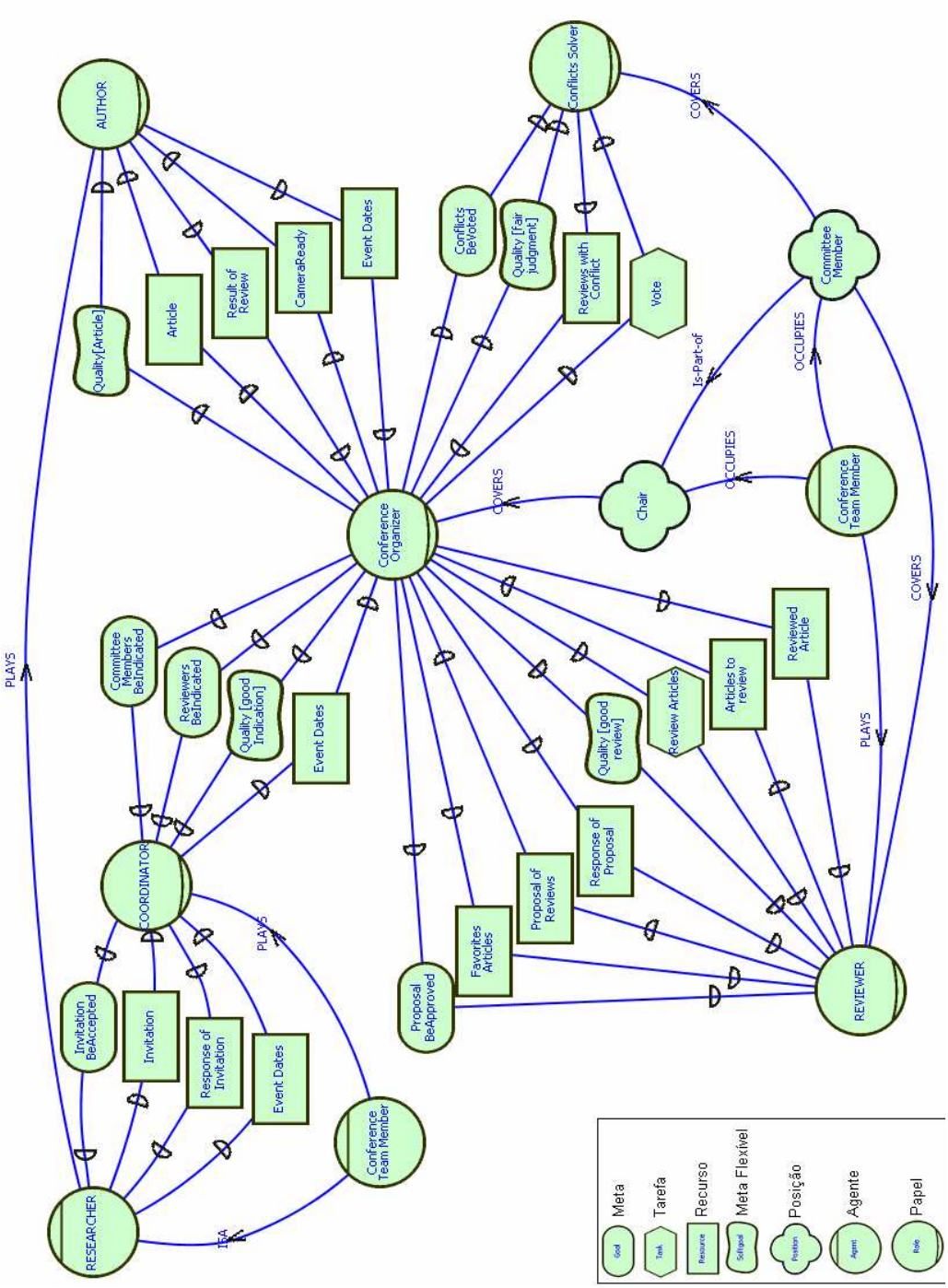

Figura 4.7 - Modelo SD Refinado

Passo 8 (Definição de requisitos gerais) – Identificação de Metas e Tarefas

Neste passo são elaborados os modelos SR com o objetivo de representar explicitamente as estratégias internas de cada ator. Os modelos elaborados apresentam, ao mesmo tempo, os SRs de dois atores que possuem interdependência estratégica entre si [17]. Os modelos SR elaborados incialmente

para o *Expert Committee* são apresentados nas figuras 4.8 a 4.12 a seguir, juntamente com uma breve descrição para cada modelo.

Modelo SR *Researcher* x *Coordinator*

No Modelo SR, apresentado na Figura 4.8, o agente 'Pesquisador' (*Researcher*) tem com principal meta 'Fazer parte da conferência' (*Join the Conference*). Esta meta pode ser alcançada através da execução da tarefa 'responder o convite' (*Response to Invitation*), que é decomposta pela sub-tarefas 'receber convite' (*Receive Invitation*), 'checar agenda' (*Check schedule*) e 'enviar resposta' (*Send Response*). A sub-tarefa 'receber convite' depende do recurso 'convite' (*Invitation*) e a sub-tarefa 'checar agenda' do recurso 'datas do evento' (*Event Dates*), recursos estes a serem disponibilizados pelo Coordenador. A sub-tarefa 'consultar agenda' é decomposta pelo recurso 'agenda pessoal' (*Personal Schedule*). A meta principal do Coordenador é que o 'time da conferência seja preenchido' (*Conference Team Be Fulfilled*) que depende externamente da meta 'convite ser aceito' (*Invitation Be Accepted*) por parte do Pesquisador. A meta pode ser alcançada através da execução da tarefa 'convidar pesquisadores' (*Invite Researchers*) que é decomposta pelas sub-tarefas 'convidar pesquisador a participar como membro do comitê de programa' (*Invite Researcher to Join as Committee Members*), 'convidar pesquisador a participar como revisor' (*Invite Researcher to Join as Reviewer*), 'divulgar datas do evento' (*Divulge Event Dates*) e 'receber resposta' (*Receive Response*). 'Convidar pesquisador a participar como revisor' é decomposta nas sub-tarefas 'preparar convites dos revisores' (*Prepare Reviewers Invitation*) e 'enviar convite' (*Send Invitation*), semelhante a 'convidar pesquisador a participar como membro do comitê de programa' que é decomposta em 'preparar convites dos membros do comitê de programa' (*Prepare Committee Members Invitation*) e 'enviar convite' (*Send Invitation*). A sub-tarefa 'receber resposta' depende externamente do recuros 'resposta ao convite' (*Response to Invitation*) a ser disponibilizado pelo Pesquisador.

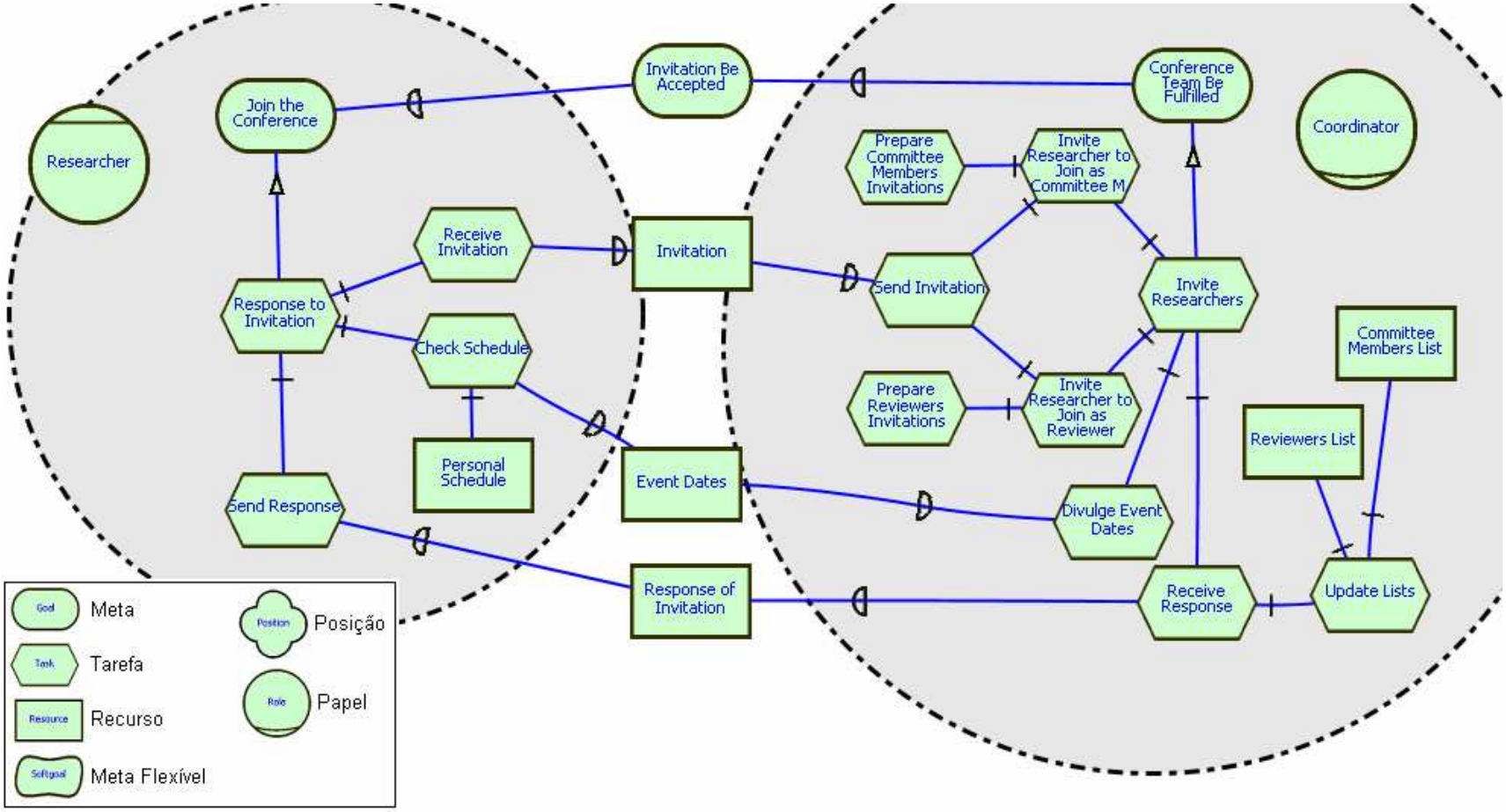

Figura 4.8 - Modelo SR: Researcher x Coodinator

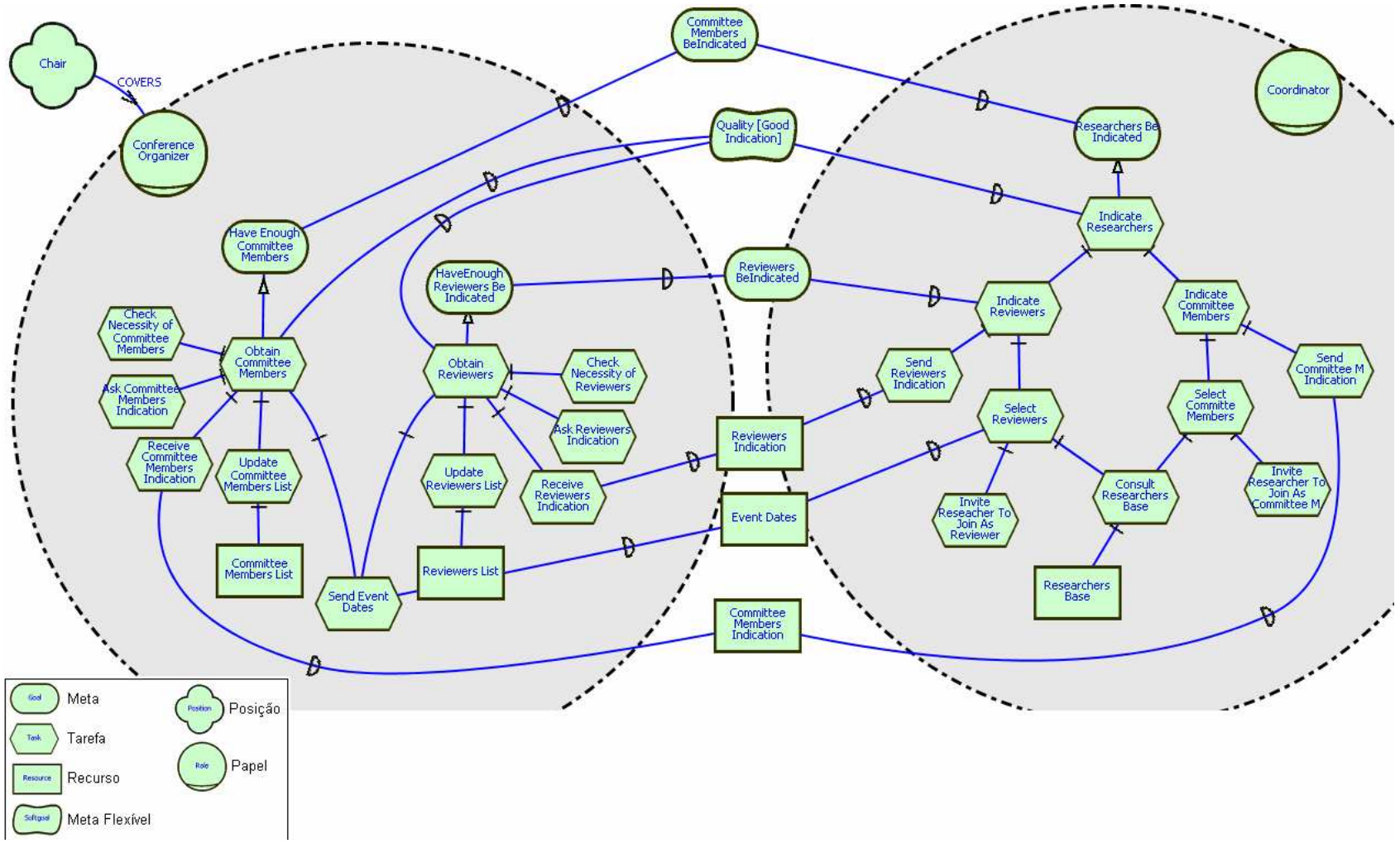

Figura 4.9 - Modelo SR: Conference Organizer (Chair) x Coordinator - modelo complementar de [17]

Modelo SR *Conference Organizer (Chair)* x *Coordinator*

No Modelo SR, apresentado na Figura 4.9, o papel de Organizador da Conferência (*Conference Organizer*), que é coberto pela posição de *Chair*, tem como metas principais ‘ter revisores em número suficiente’ (*Have Enough Reviewers*) e ‘ter membros do comitê em número suficiente’ (*Have Enougn Committee Members*). A meta ‘ter revisores em número suficiente’ depende da meta externa ‘revisores serem indicados’ (*Reviewers Be Indicated*) e é alcançada através da tarefa ‘obter revisores’ (*Obtain Reviewers*). A tarefa ‘obter revisores’ é decomposta pelas sub-tarefas ‘checar necessidade de revisores’ (*Check Necessity of Reviewers*), ‘solicitar indicação de revisores’ (*Ask Reviewers Indication*), ‘receber indicação de revisores’ (*Receive Reviewers Indication*), ‘atualizar lista de revisores’ (*Update Reviewers List*) e ‘enviar datas do evento’ (*Send Event Dates*). A sub-tarea ‘receber indicação de revisores’ dependo do recurso externo ‘indicação de revisores’ (*Reviewers Indication*) a ser disponibilizado pelo Coordenador. A qualidade das indicações, representada pela meta flexível ‘qualidade – boa indicação’ (*Quality [Good Indication]*) que o Organizador da conferência recebe é uma dependência externa do Coordenador. A meta ‘ter membros do comitê em número suficiente’ é decomposta de maneira similar a meta ‘ter revisores em número suficiente’.

O papel de Coordenador (*Coordinator*) tem como meta principal que ‘pesquisadores sejam indicados’ (*Researchers Be Indicated*). Esta meta é alcançada através da tarefa ‘indicar pesquisadores’ (*Indicate Researchers*), que é decomposta em duas sub-tarefas: ‘indicar revisores’ (*Indicate Reviewers*) e ‘indicar membros do comitê de programa’ (*Indicate Committee Members*). A sub-tarefa ‘indicar revisores’ é decomposta pelas sub-tarefas ‘selecionar revisores’ (*Select Reviewers*) e ‘enviar indicação de revisores’ (*Send Reviewers Indication*). A sub-tarefa ‘selecionar revisores’ depende externamente do recurso ‘datas do evento’ e é decomposta nas sub-tarefas ‘consultar base de pesquisadores’ (*Consult Researchers Base*) e ‘convidar pesquisador a participar como revisor’ (*Invite Researcher to Join as Reviewer*). A sub-tarefa ‘consultar base de pesquisadores’ é decomposta pelo recurso ‘base de pesquisadores’ (*Researchers Base*). A tarefa

‘indicar membros de comitê de programa’ é decomposta de modo similar a tarefa ‘indicar revisores’ conforme exposto no modelo.

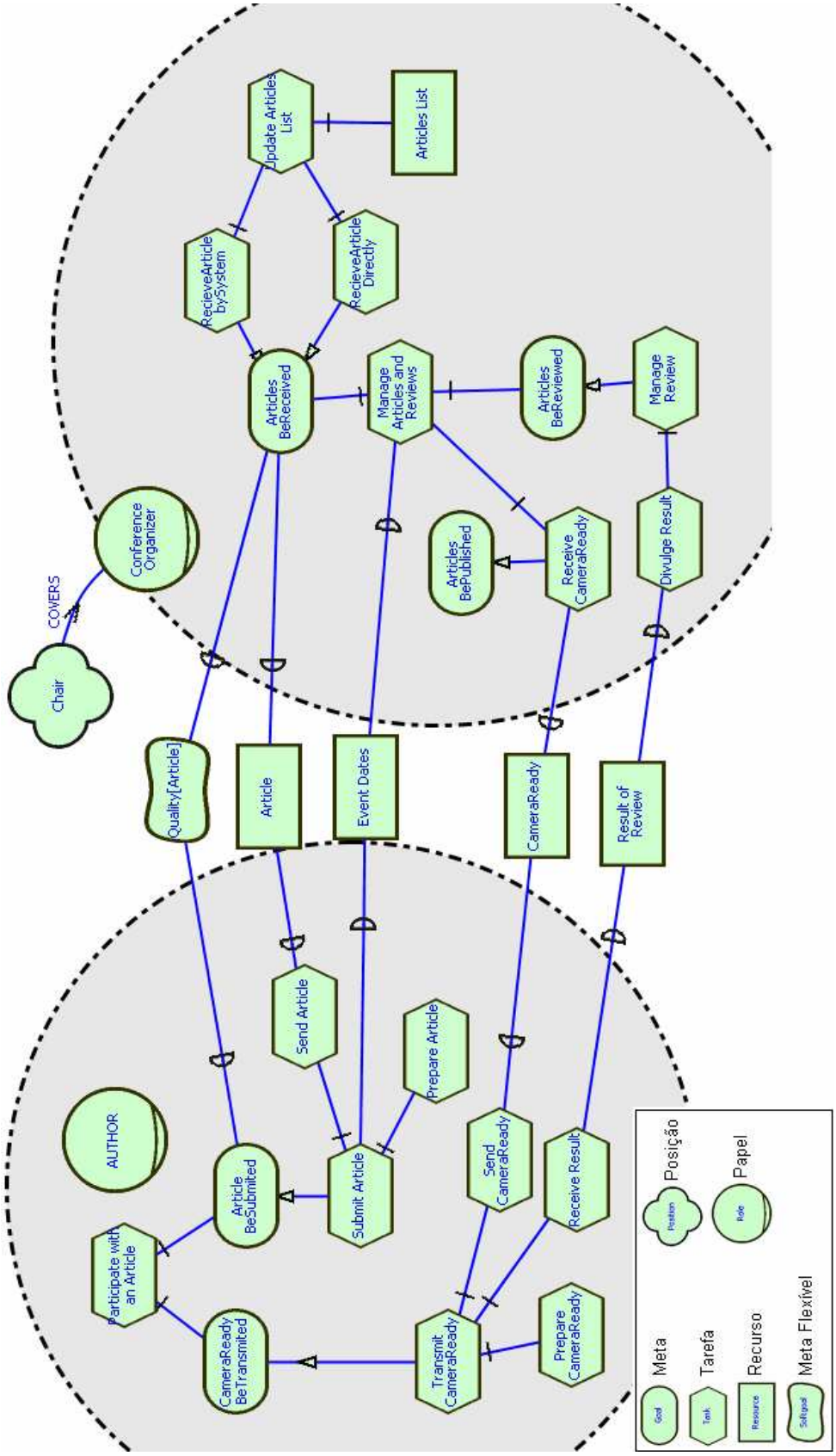

Figura 4.10 - Modelo SR: Author x Conference Organizer (Chair) - modelo complementar de [17]

Modelo SR *Author x Conference Organizer (Chair)*

No Modelo SR, apresentado na Figura 4.10, o papel de 'Autor' (*Author*) tem como metas principais 'artigo ser submetido' (*Article Be Submited*) e 'versão final ser transmitida' (*CameraReady Be Transmited*). A meta 'artigo ser submetido' é alcançada através da tarefa 'submeter artigo' (*Submit Article*), que é decomposta pelas sub-tarefas 'preparar artigo' (*Prepare Article*) e 'submeter artigo' (*Submit Article*) e depende externamente do recurso 'datas do evento' (*Event Dates*).

A meta 'versão final ser transmitida' é alcançada através da tarefa 'transmitir versão final' (*Transmit CameraReady*) que é decomposta pelas sub-tarefas 'receber resultado' (*Receive Result*), que possui dependência externa para o recurso 'resultado da revisão' (*Result of Review*), 'prepara revisão' (*Prepare Review*) e 'enviar revisão' (*Send Review*).

O papel de 'Organizador da Conferência', coberto pela posição de *Chair*, tem como meta principal que 'artigos sejam publicados' (*Articles Be Published*). Para alcançar esta meta, é necessário executar a tarefa 'receber versão final' (*Receive CameraReady*), que depende do recurso externo 'versão final' (*CameraReady*) e é decomposta pela sub-tarefa 'gerenciar artigos e revisões' (*Manage Articles and Reviews*). 'Gerenciar artigos e revisões' é decomposta pelas sub-metas 'artigos serem recebidos' (*Articles Be Received*) e 'artigos serem revisados' (*Articles Be Reviewed*). A sub-meta 'artigos serem recebidos' pode ser alcançada por uma das tarefas alternativas: 'receber artigo pelo sistema' (*Receive Article by System*) ou 'receber artigo diretamente' (*Receive Article Directly*). Ambas são decompostas pela sub-tarefa 'atualizar lista de artigos' (*Update Articles List*) que por sua vez é decomposta pelo recurso 'lista de artigos' (*Articles List*). A meta 'artigos serem revisados' é alcançada através da tarefa 'gerenciar revisões' (*Manage Review*) que é decomposta pela sub-tarefa 'divulgar resultado' (*Divulge Result*). 'Gerenciar revisões' é decomposta também por outras tarefas e recursos, o que é abordado no 'Modelo SR *Reviewer x Conference Organizer (Chair)*', apresentado na Figura 4.11.

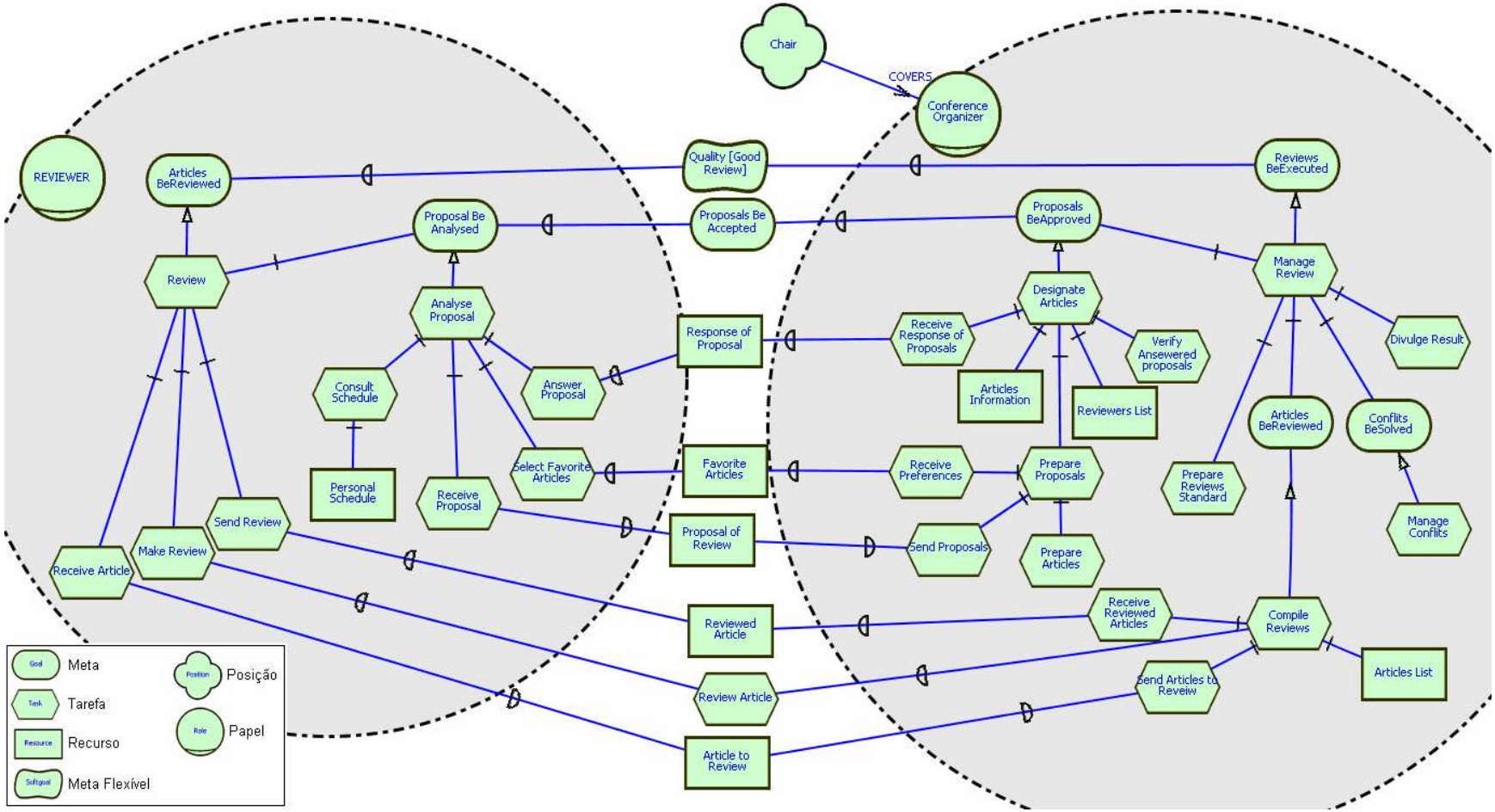

Figura 4.11 - Modelo SR: Reviewer x Conference Organizer (Chair) - modelo adaptado de [17]

Modelo SR *Reviewer x Conference Organizer (Chair)*

No Modelo SR, apresentado na Figura 4.11, o papel de Revisor (*Reviewer*) tem como meta principal que os 'artigos sejam revisados' (*Articles Be Reviewed*). Esta meta é alcançada via execução da tarefa 'revisar' (*Review*). 'Revisar' por sua vez é decomposta pela meta 'proposta ser analisada' (*Proposal Be Analysed*) e pelas sub-tarefas 'receber artigo' (*Receive Article*), 'fazer revisão' (*Make Review*) e 'enviar revisão' (*Send Review*). A tarefa 'receber artigo' (*Receive Article*) depende de um recurso externo 'artigo para revisão' (*Article to Review*) a ser disponibilizado pelo papel do Organizador da Conferência (*Conference Organizer*). A meta 'proposta ser analisada' é alcançada pela execução da tarefa 'analisar proposta' (*Analyse Proposal*) que por sua vez é decomposta nas sub-tarefas 'receber proposta' (*Receive Proposal*), 'consultar agenda' (*Consult Schedule*), 'selecionar artigos favoritos' (*Select Favorites Articles*) e 'responder proposta' (*Response Proposal*). A sub-tarefa 'receber proposta' depende de um recurso externo 'proposta de revisão' (*Proposal of Review*) a ser disponibilizado pelo Organizador da Conferência, enquanto a sub-tarefa 'consultar agenda' é decomposta no recurso 'agenda pessoal' (*Personal Schedule*).

O papel de Organizador da Conferência, coberto pela posição de *Chair*, tem como meta principal que as 'revisões sejam executadas' (*Reviews Be Executed*). Para que esta meta seja alcançada é necessário executar a tarefa 'gerenciar revisões' (*Manage Reviews*).

A tarefa 'gerenciar revisões' decomposta pelas sub-metas 'propostas serem aprovadas' (*Proposasl Be Approved*), 'artigos serem revisados' (*Articles Be Reviewed*), 'conflitos serem resolvidos' (*Conflicts Be Solved*) e pelas sub-tarefa 'preparar padrões para revisão' (*Prepare Reviews Standard*) e 'divulgar resultado' (*Divulge Result*). A sub-meta 'artigos serem revisados' é alcançada através da execução da tarefa 'compilar revisões' (*Compile Reviews*). 'Compilar revisões', por sua vez, é decomposta pelas sub-tarefas 'enviar artigos para revisão' (*Send Articles to Review*), 'receber artigos revisados' e pelo recurso 'lista de artigos' (*Articles List*) e depende externamente da execução da tarefa 'revisar artigo' pelo Revisor. A meta 'conflitos serem resolvidos' (*Conflicts Be Solved*) é alcançada pela execução da tarefa 'gerenciar conflitos' (*Manage Conflicts*) que será

detalhada no Modelo SR ‘Conflicts Solver (Committee Member) x Conference Organizer (Chair)’, apresentado na Figura 4.12.

A meta ‘propostas serem aprovadas’ é alcançada pela execução da tarefa ‘designar artigos’ (*Designate Articles*) e depende externamente da meta ‘propostas serem aceitas’. A tarefa ‘designar artigos’ é decomposta pelos recursos ‘informações dos artigos’ (*Articles Informations*) e ‘lista de revisores’ (*Reviewers List*) e pelas sub-tarefas ‘preparar propostas’ (*Prepare Proposals*), ‘receber resposta de propostas’ (*Receive Response of Proposals*) que possui uma dependência externa do recurso ‘resposta de proposta’ (*Response of Proposal*) e ‘verificar propostas respondidas’ (*Verify Answered Proposals*). ‘Prepara propostas’ se decompõe em três sub-tarefas: ‘receber preferências’ (*Receive Preferences*) que depende externamente do recurso ‘artigos favoritos’ a ser disponibilizado pelo Revisor, ‘preparar artigos’ (*Prepare Articles*) e ‘enviar propostas’ (*Send Proposals*).

Modelo SR *Conflicts Solver (Committee Member) x Conference Organizer* (*Chair*)

No Modelo SR, apresentado na Figura 4.12, o papel de ‘Resolvedor de conflitos’ (*Conflicts Solver*), coberto pela posição de ‘membro do comitê de programa’ (*Committee Member*), tem como meta principal que os ‘conflitos sejam julgados’ (*Conflicts Be Judged*). Esta meta é alcançada através da tarefa ‘julgar conflitos em revisões’ (*Judge Conflicts in Reviews*) que é decomposta nas sub-tarefas ‘receber revisões com conflito’ (*Receive Reviews with Conflict*) e na meta-flexível ‘formar uma opinião’ (*Make Up his Mind*), que foi modelada como tal por ser uma meta subjetiva. A tarefa ‘receber revisões com conflito’ depende externamente do recurso ‘revisões com conflito’ (*Reviews with Conflict*) a ser disponibilizado pelo Organizador da conferência.

O papel de Organizador da conferência, coberto pela posição de *Chair*, tem como meta principal que os ‘conflitos sejam resolvidos’ (*Conflicts Be Solved*). Esta meta depende externamente da meta ‘conflitos serem votados’ (*Conflicts Be Voted*) e da meta flexível ‘qualidade – julgamento justo’ (*Quality [Fair Judgement]*), e internamente a tarefa ‘resolver conflitos’ (*Solve Conflicts*) contribui para seu alcance.

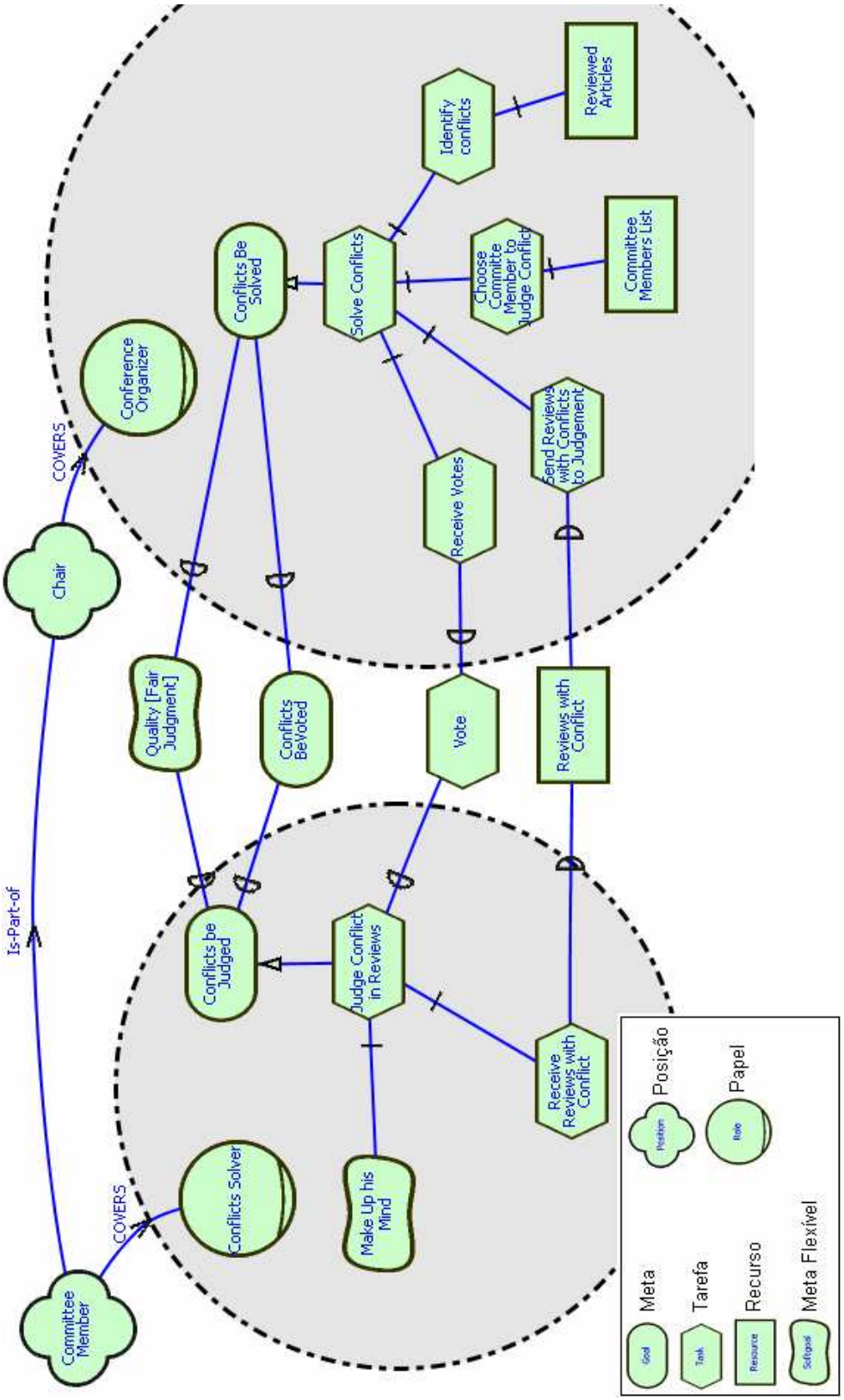

Figura 4.12 - Modelo SR: Conflicts Solver (Committee Member) x Conference Organizer (Chair) - modelo complementar de [17]

A tarefa ‘Resolver conflitos’ é decomposta em ‘identificar conflitos’ (*Identify Conflicts*), ‘escolher membro do comitê para julga conflito’ (*Choose Committee Member to Judge Conflict*), ‘enviar revisões com conflito para julgamento’ (*Send*

Reviews with Conflict to Judgement) e ‘receber votos’ (*Receive Votes*). ‘Receber votos’ depende externamente da execução da tarefa ‘votar’ (*Vote*) pelo ‘Resolvedor de Conflitos’.

Passo 3 (Definição de requisitos específicos de segurança) – Identificação de intenções maliciosas

Neste passo, são identificadas intenções maliciosas dos atacantes, identificados anteriormente no passo 1 (Definição de requisitos específicos de segurança) – identificação de atacantes. Intenções maliciosas são basicamente intenções prejudiciais à segurança, que os atacantes teriam, ou têm, ao atacar o sistema. Esta análise usa também as informações contidas nos modelos SR, elaborados no passo 8 (definição de requisitos gerais), uma vez que os atacantes poderiam se apropriar da capacidade dos atores legítimos para executar ações ilegais ou acessar recursos valiosos.

A Figura 4.13, a seguir, apresenta em um modelo SR parcial (composto apenas por metas) as intenções maliciosas dos atacantes internos. Neste modelo, o agente ‘Atacante do Pesquisador’ tem como intenção maliciosa ‘Pesquisador legítimo ser desacreditado’, que é representada pela meta ‘*Legitimate Researcher Be Discredited*’. O agente ‘Atacante do Autor’ (*Author Attacker*) tem como intenções maliciosas ‘Autor legítimo ser desacreditado’, ‘artigos serem roubados’ e ‘artigos serem fraudados’, representadas respectivamente pelas metas ‘*Legitimate Author Be Discredited*’, ‘*Articles Be Stolen*’ e ‘*Articles Be Defrauded*’ respectivamente. O agente ‘Atacante do Revisor’ (*Reviewer Attacker*) tem como intenções maliciosas ‘Revisor legítimo ser desacreditado’, ‘revisões serem roubadas’ e ‘revisões serem fraudadas’, representadas respectivamente pelas metas ‘*Legitimate Reviewer Be Discredited*’, ‘*Reviews Be Stolen*’ e ‘*Reviews Be Defrauded*’ respectivamente. O agente ‘Atacante do Membro do comitê’ (*Committee Member Attacker*) tem como intenções maliciosas ‘Membro do comitê legítimo ser desacreditado’, ‘revisões serem roubadas’ e ‘votos serem fraudados’, representadas respectivamente pelas metas ‘*Legitimate Committee Member Be Discredited*’, ‘*Reviews Be Stolen*’ e ‘*Votes Be Defrauded*’ respectivamente. O agente ‘Atacante do Coordenador’ (*Coordinator Attacker*) tem como intenções maliciosas ‘Coordenador legítimo ser desacreditado’ e ‘más

indicações serem feitas', representadas respectivamente pelas metas '*Legitimate Coordinator Be Discredited*' e '*Bad Reviews Be Done*' respectivamente. O agente 'Atacante do *Chair*' tem como intenções maliciosas '*Chair* legítimo ser desacreditado', 'conferência ser desacreditada', 'artigos serem roubados', 'artigos serem fraudados', 'revisões serem roubadas', 'revisões serem fraudadas' e 'votos serem fraudados' representadas respectivamente pelas metas '*Legitimate Chair Be Discredited*', '*Conference Be Discredited*', '*Articles Be Stolen*', '*Articles Be Defrauded* , '*Reviews Be Stolen*', '*Reviews Be Defrauded*' e '*Votes Be Defrauded*' respectivamente.

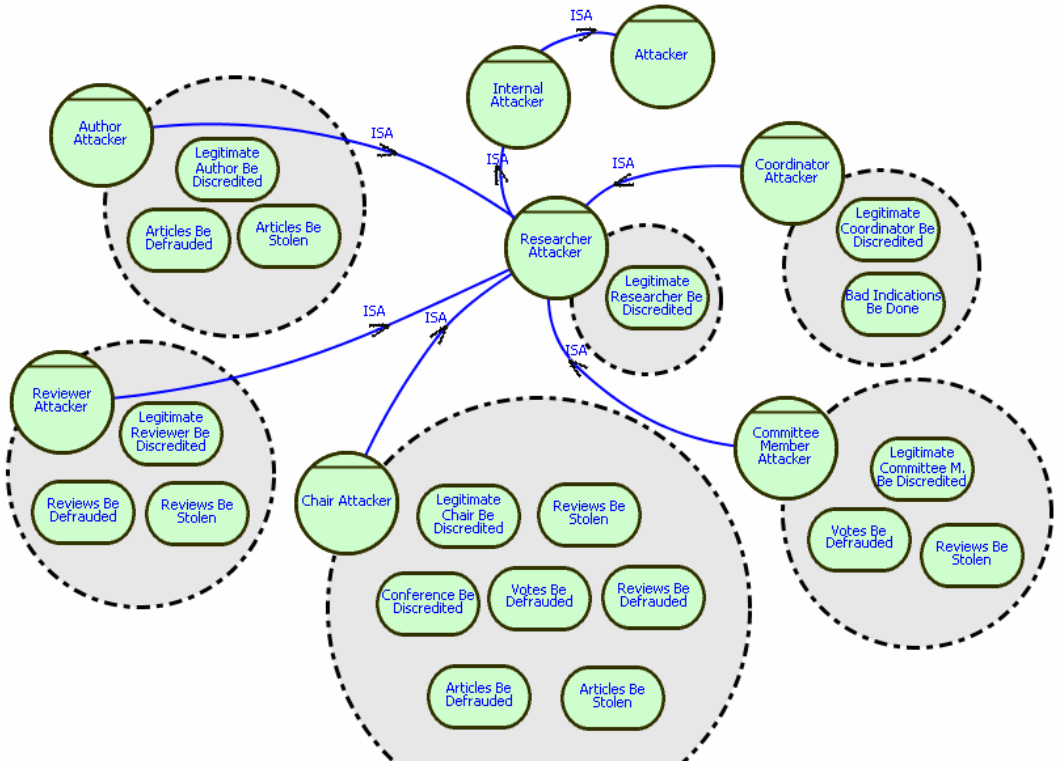

Figura 4.13 - Intenções maliciosas dos atacantes internos

A Figura 4.14, a seguir, apresenta o modelo SR parcial com as intenções maliciosas dos atacantes externos. Neste modelo, o agente 'Atacante externo' (*External Attacker*) tem como intenções maliciosas 'informações valiosas serem roubadas' e 'conferência ser desacreditada', representadas respectivamente pela metas '*Valuable Informations Be Stolen*' e '*Conference Be Defrauded*'.

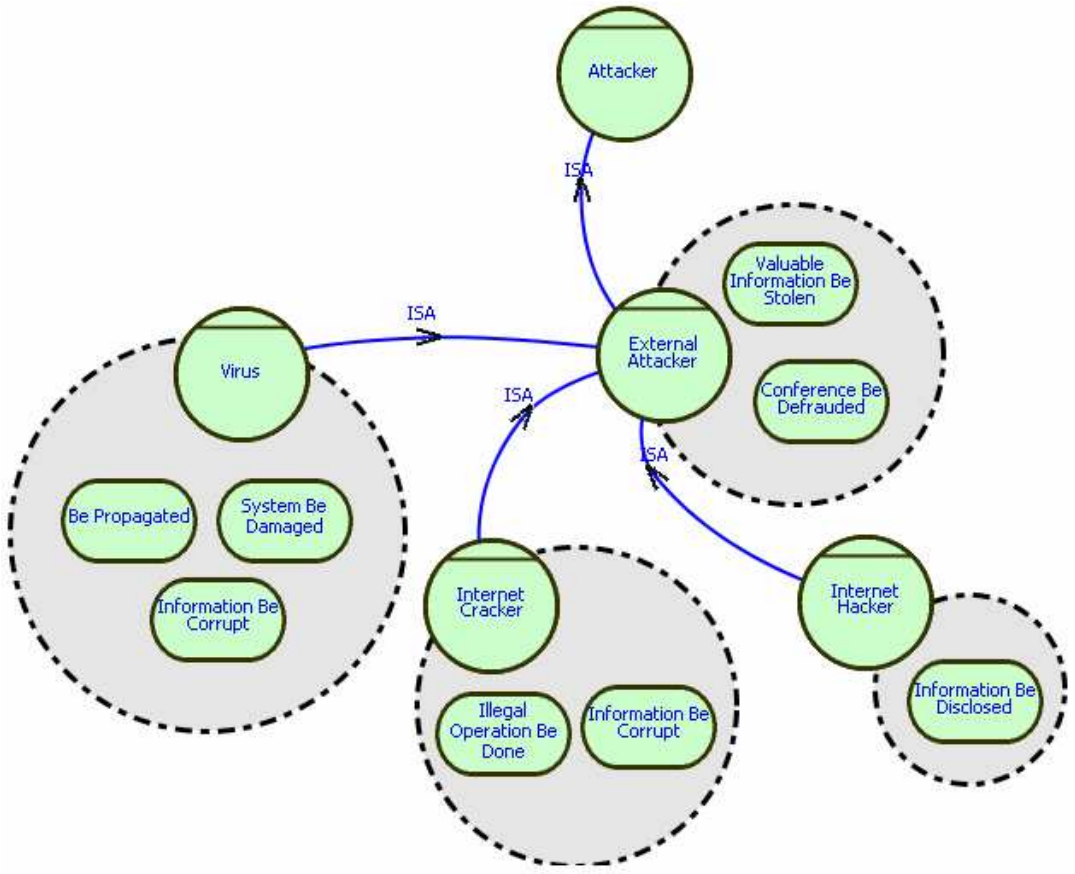

Figura 4.14 - Intenções maliciosas dos atacantes externos

O agente ‘Virus’ tem como intenções maliciosas ‘ser propagado’, ‘informação ser corrompida’ e ‘sistema ser danificado’, que são representadas respectivamente pelas metas ‘*Be Propagated*’, ‘*Information Be Corrupt*’ e ‘*System Be Damaged*’. O agente ‘Cracker’ (*Internet Cracker*) tem como intenções maliciosas ‘operações ilegais serem realizadas’ e ‘informações serem corrompidas’, representadas respectivamente pelas metas ‘*Illegal Operation Be Done*’ e ‘*Information Be Corrupt*’. O agente ‘*Hacker*’ (*Internet Hacker*) tem como intenção maliciosa ‘*Informações serem descobertas*’, representada pela meta ‘*Information Be Disclosed*’.

Os modelos apresentados não são exaustivos em relação às intenções maliciosas. É bastante difícil se verificar a completeza na identificação das intenções dos atacantes.

Passo 9 – iterativo (Definição de requisitos gerais) – Refinamento de Cenários

Neste passo os cenários, elaborados anteriormente no passo 5 – Definição de cenários, são refinados com base nos modelos SR elaborados no passo 8 – Identificação de metas e tarefas. As linhas acrescentadas aos cenários estão em azul.

Scenario definition		
Title:	**Researchers Invitation** (incomplete)	
Goal:	Invitation be accepted.	
Context:		
Geographical Location:	WEB	
Temporal Location:	Before the beginning the article submission phase.	
Constraint:		
Precondition:		
Resources:	Computer, Internet, event dates, researchers personal schedule	
Constraint:		
Actors:	Coordinator, Researcher	
Episodes:	Coordinator Divulge Event Dates	
	Coordinator Invite Researchers	
	Coordinator Invite Reviewers	
	Coordinator prepares the reviewers invitations	
	Coordinator sends the invitations to Researchers	
	Coordinator Invite Committee Members	
	Coordinator Invite Committee Members	
	Coordinator prepares the committee members invitations	
	Coordinator sends the invitations to Researchers	
	Researcher receive invitation	
	Researcher check his schedule	
	Researcher sends the response of invitation to Coordinator	
	Coordinator receives the responses	
	Coordinator update the reviewers list and the committee members list	
Constraint:	NFR: Security, User-friendliness	
Exceptions:		

Scenario definition		
Title:	**Reviewers Indication** (incomplete)	
Goal:	Reviewers be indicated.	
Context:		
Geographical Location:	WEB	
Temporal Location:	Until the reviews deadline.	
Constraint:		
Precondition:	Invitation Acceptance	
Resources:	Computer, Internet	
Constraint:		
Actors:	Chair, Coordinator	
Episodes:	Chair check the necessity of Reviewers	
	Chair ask Coordinator for Reviewers indication (specifying the areas)	
	Chair informs the event dates to Coordinator.	
	Coordinator prepare the Reviewers indication (according to the specified areas)	
	Coordinator selects reviewers	
	Coordinator consults researchers base	
	Coordinator invite researcher (scenario: Researchers Invitation)	
	Coordinator send the Reviewers indication to Chair	
	Chair receives the Reviewers indication.	
	Chair update the Reviewers list	
Constraint:	NFR: Security, User-friendliness	
Exceptions:		

Scenario definition		
Title:	**Committee Members Indication** (incomplete)	
Goal:	Committee Members be indicated	
Context:		
Geographical Location:	WEB	
Temporal Location:	Until the reviews deadline.	
Constraint:		
Precondition:	Invitation Acceptance	
Resources:	Computer, Internet	
Constraint:		
Actors:	Chair, Coordinator	
Episodes:	Chair check the necessity of Committee Members	
	Chair ask Coordinator for Committee Members indication (specifying the areas)	
	Chair informs the event dates to Coordinator.	
	Coordinator prepare the Committee Members indication (according to the specified areas)	
	Coordinator selects committee members	
	Coordinator consults researchers base	
	Coordinator invite researcher (scenario: Researchers Invitation)	
	Coordinator send the Reviewers indication to Chair	
	Chair receives the Committe Members indication.	
	Chair updates the Committee Members list	
Constraint:	NFR: Security, User-friendliness	
Exceptions:		

Scenario definition		
Title:	**Article Submission** (incomplete)	
Goal:	Article be submitted.	
Context:		
Geographical Location:	WEB	
Temporal Location:	Before the submission deadline.	
Constraint:		
Precondition:		
Resources:	Computer, Internet, Article	
Constraint:		
Actors:	Chair, Author	
Episodes:	Chair informs the event dates	
	Author prepare the article.	
	Author sends the article to Chair.	
	Chair receives article	
	Chair receives article by System or Chair receives article directly	
	Chair updates the Articles List	
Constraint:	NFR: Security, User-friendliness	
Exceptions:		

Scenario definition		
Title:	**Proposal Acceptance** (incomplete)	
Goal:	Proposal be accepted	
Context:		
Geographical Location:	WEB	
Temporal Location:	Right after the submission deadline.	
Constraint:		
Precondition:	Reviewers list and articles list have been prepared	
Resources:	Computer, Internet, reviewers list, articles list	
Constraint:		
Actors:	Chair, Reviewers	
Episodes:	Chair receive preferences from reviewers	
	Chair prepare proposals	
	Chair selects reviewers within the same area of the article.	
	Chair separates out reviewers of the same institution of the author.	
	Chair makes proposals to reviewers.	
	Chair sends proposals to each *reviewer giving the acceptance deadline.	
	Reviewer receives proposal	
	Reviewer consults personal schedule	
	Reviewer selects favorites articles	
	Reviewer answers the proposal	
	Chair receives answered proposals form reviewers	
	Chair verifies answered proposals.	
Constraint:	NFR: Security, User-friendliness	
	* Each reviewer can not receive more than 3 articles.	
Exceptions:	If there is at least one article without 3 reviewers:	
	(scenario: “Reviewers Indication”)	

Scenario definition		
Title:	**Articles Review** (incomplete)	
Goal:	ArticlesBeReviewed	
Context:		
Geographical Location:	WEB	
Temporal Location:	Before deadline to send reviews.	
Constraint:		
Precondition:	Proposal Acceptance.	
Resources:	Computer, Internet, Articles to review	
Constraint:		
Actors:	Chair, Reviewers	
Episodes:	Chair prepares standards to review	
	Chair selects an article to sends to review	
	Chair selects reviewers that have accepted the proposal of review for the selected article;	
	Chair sends the article to each reviewer selected (in previous step)	
	Reviewer receives the article sent by Chair	
	Reviewer makes the article review	
	Reviewer sends reviewed article to Chair	
	Chair receives the reviewed articles	
	Chair checks if there is a conflict (incompatible evaluations) in the reviews of each article	
Constraint:	NFR: Security, User-friendliness	
Exceptions:	If an article has a conflict:	
	(scenario: Conflict Resolution)	

Scenario definition		
Title:	**Conflict Solution** (incomplete)	
Goal:	Conflict be solved.	
Context:		
Geographical Location:	WEB	
Temporal Location:	After articles have been reviewed.	
Constraint:		
Precondition:	Review Article	
Resources:	Computer, Internet, Articles, Reviews	
Constraint:		
Actors:	Chair, Committee Members	
Episodes:	Chair identify an article with a conflict (incompatible evaluations)	
	Chair selects Committee Members in the same area of the article.	
	Chair separates out reviewers of the same institution of the Author.	
	Chair sends reviews with conflict to each selected Committee Member giving the deadline to vote	
	Committee Members receives the reviews with conflict from Chair	
	Committee Members votes (to solve the conflict)	
	Chair receives the votes	
Constraint:	NFR: Security, User-friendliness	
Exceptions:		

Scenario definition		
Title:	**Camera Ready Reception** (incomplete)	
Goal:	Camera Ready be sent.	
Context:		
Geographical Location:		WEB
Temporal Location:		Before the camera ready sending deadline.
Constraint:		
Precondition:		Article be accepted
Resources:	Computer, Internet, camera ready	
Constraint:		
Actors:	Author, Chair	
Episodes:	Chair sends the result of review to the Author.	
	Author receives the result of review	
	Author prepares the camera ready.	
	Author sends the camera ready to Chair.	
	Chair receives the camera ready	
Constraint:		NFR: Security, User-friendliness
Exceptions:		

Passo 10 (Definição de requisitos gerais) – Definição de Requisitos Não-Funcionais

Neste passo são definidos os requisitos não-funcionais. Esta definição é feita em duas etapas: (i) criação de metas flexíveis para os requisitos não-funcionais e (ii) refinamento dos requisitos não funcionais.

(i) Criação de metas flexíveis para os requisitos não-funcionais – nesta etapa são adicionados aos modelos SR metas flexíveis correspondentes aos requisitos não-funcionais.

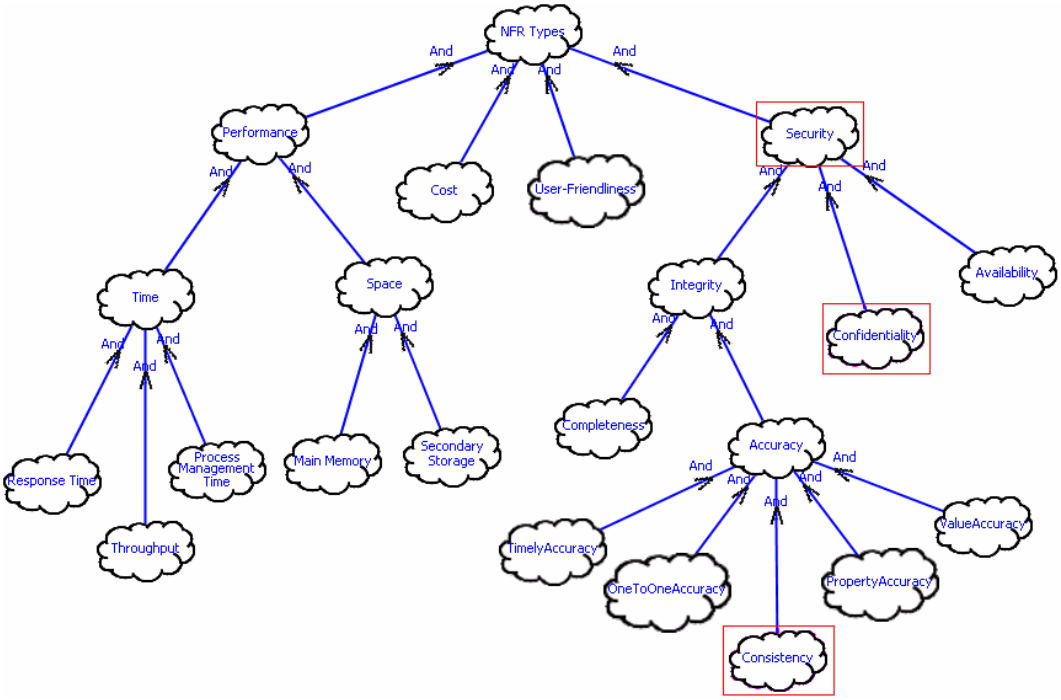

Figura 4.15 - Tipos de requisitos não-funcionais, baseado em [7]

A elicitação dos requisitos não-funcionais, e seus refinamentos, será feita utilizando-se um catálago de requisitos não-funcionais como proposto por Chung et al [7]. A Figura 4.15 apresenta tipos de requisitos não-funcionais. Os requisitos de nível de abstração mais alto são decompostos em sub-tipos de nível de abstração menor via elos de contribuição. Como o foco deste estudo de caso são requisitos de segurança, especificamente confidencialidade e consistência das informações, as metas flexíveis relativas aos demais requisitos não-funcionais não serão abordadas. Serão criadas apenas metas flexíveis de alto nível para alguns requistos não-funcionais que sejam impactados, ou impactem, os requistos de segurança abordados, como por exemplo os requisitos não-funcionais de usabilidade e desempenho. Com o auxílio de um catálogo, os requisitos de segurança são identificados com o objetivo de neutralizar as vulnerabilidades identificadas anteriormente no passo 2 – identificação de vulnerabilidades (tabela 4.5).

(ii) refinamento dos requisitos não funcionais – uma meta flexível pode ser refinada por tipo e por tópico. No refinamento por tipo, o requisito não-funcional (meta flexível) é decomposto em sub-tipos (sub-metas).

No refinamento por tópicos, o requisito não-funcional é decomposto por tópicos (ou assuntos). Neste caso, o requisito não-funcional é aplicado a cada um dos tópicos do refinamento. Tanto no refinamento por tipos quanto no refinamento por tópicos, a idéia é que os requisitos de mais alto nível serão 'satisfeitos a contento' com a 'satisfação a contento' dos requisitos que os refinam. O refinamento por tipos independe do domínio da aplicação, o que permite que seja facilmente reusado. Já no refinamento por tópicos, são usados tópicos pertencentes ao domínio que se deseja modelar, o que pode restingir sua reutilização.

A Figura 4.16, a seguir, apresenta o refinamento por tipo do requisito não-funcional de segurança, com destaque para confidencialidade e consistência, foco deste trabalho.

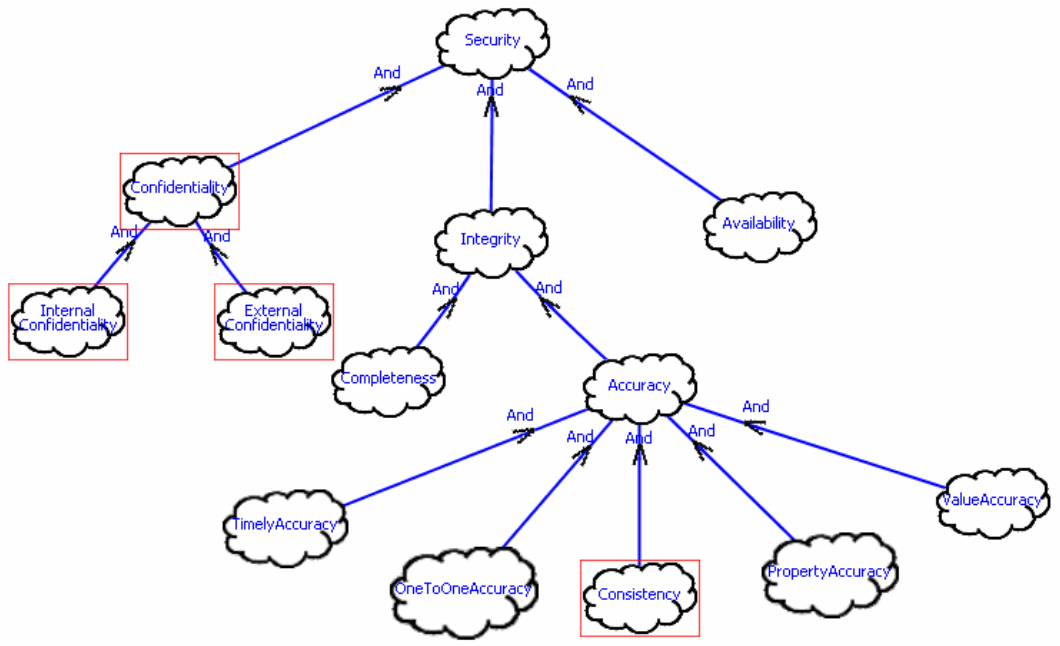

Figura 4.16 - Refinamento de segurança por tipo, baseado em [7]

O requisito de confidencialidade (*Confidentiality*) está decomposto em dois tipos: confidencialidade interna e confidencialidade externa. Confidencialidade interna significa que deve ser respeitada a confidencialidade entre os atores internos à conferência, representados no modelo SA, Figura 4.5, pelo agente Membro da

equipe da conferência. Já confidencialidade externa, significa que o acesso às informações é vetado apenas aos atores externos à conferência.

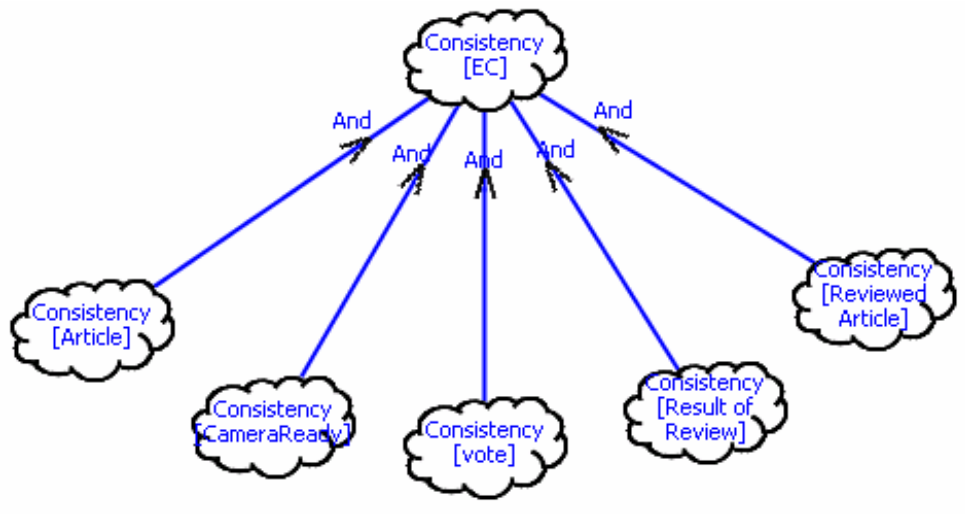

Figura 4.17 - Refinamento de consistência por tópicos.

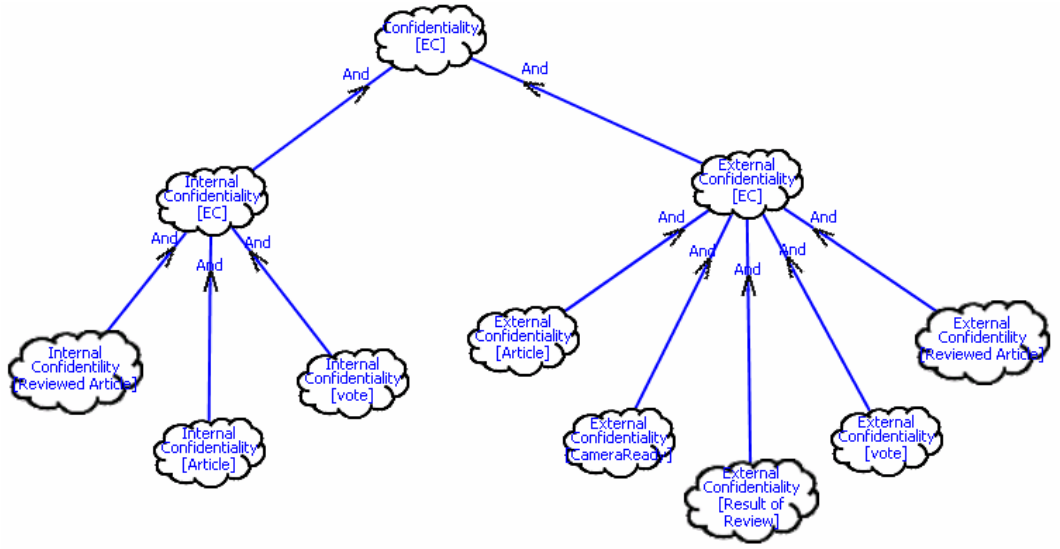

Figura 4.18 - Refinamento de confidencialidade por tópicos

A Figura 4.17 apresenta o refinamento do requisito de consistência por tópicos e a Figura 4.18 apresenta o refinamento de confidencialidade, e seus sub-tipos confidencialidade interna e confidencialidade externa, por tópicos.

Passo 11 (iterativo) – Adição dos requisitos não funcionais aos modelos SR

Neste passo, os requisitos não-funcionais, refinados por tipos e tópicos no passo anterior, são adicionados aos modelos SR produzidos no passo 8 (Identificação de metas e tarefas), através de metas flexíveis. Por uma questão de simplicidade, as metas flexíveis de segurança serão adicionadas as tarefas de nível de abstração mais alto associadas aos recursos vulneráveis, identificados no passo 2 (definição de requisitos específicos de segurança) – Identificação de vulnerabilidades. Uma meta flexível adicionada a uma tarefa de nível de abstração mais alto desdobra-se para todas as sub-tarefas que decompõe esta tarefa em níveis de abstração menores. As figura 4.19, 4.20 e 4.21 apresentam os modelos SR com metas flexíveis correspondentes aos requisitos de confidencialidade e consistência refinados por tipos e tópicos. Tanto as metas flexíveis quanto as tarefas impactadas estão destacadas por um retângulo vermelho nos modelos.

Com o refinamento dos modelos SR pela adição das metas flexíveis, torna-se necessário também refinar os cenários correspondentes, através de uma nova execução do passo 9 (iterativo).

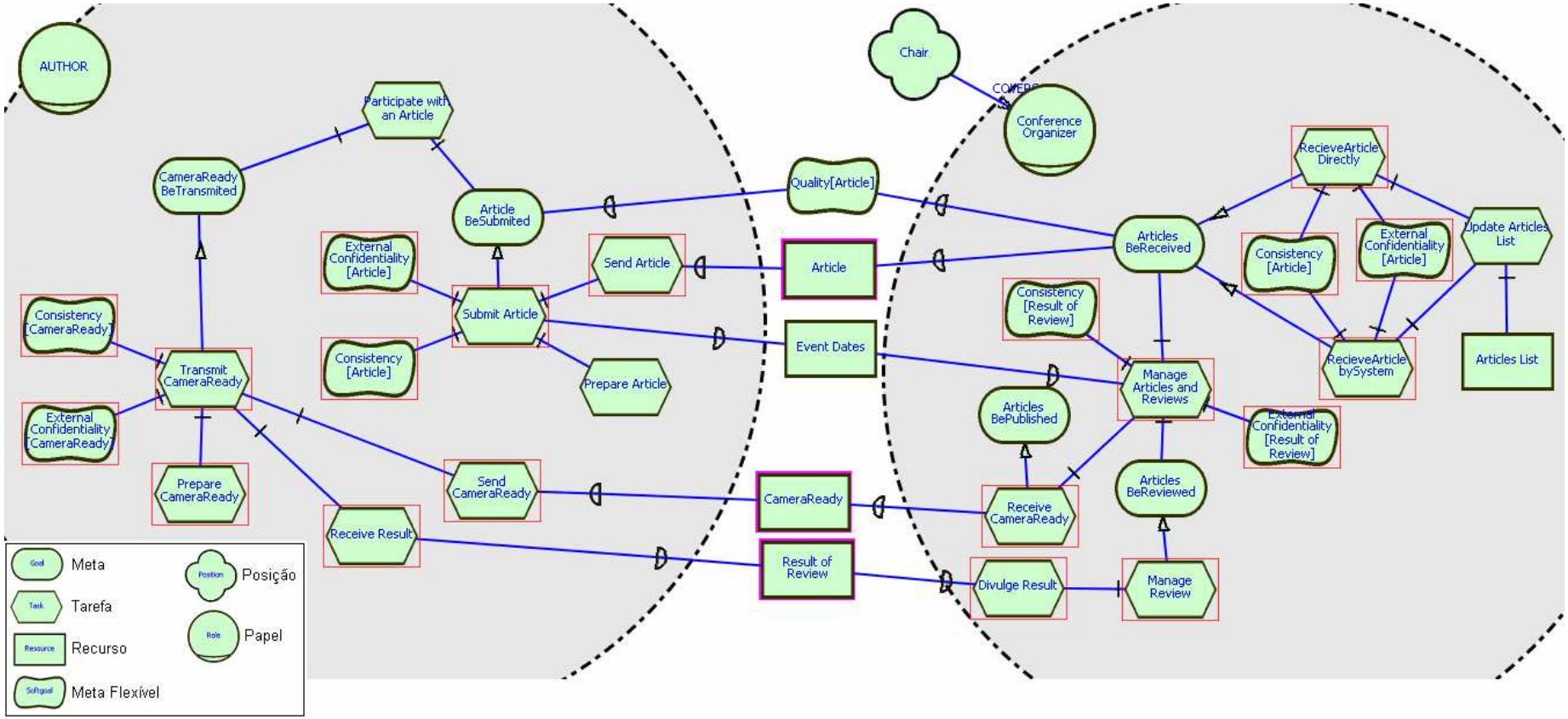

Figura 4.19 - Adição de metas flexíveis de segurança às situações de submissão de artigo e recepção de versão final

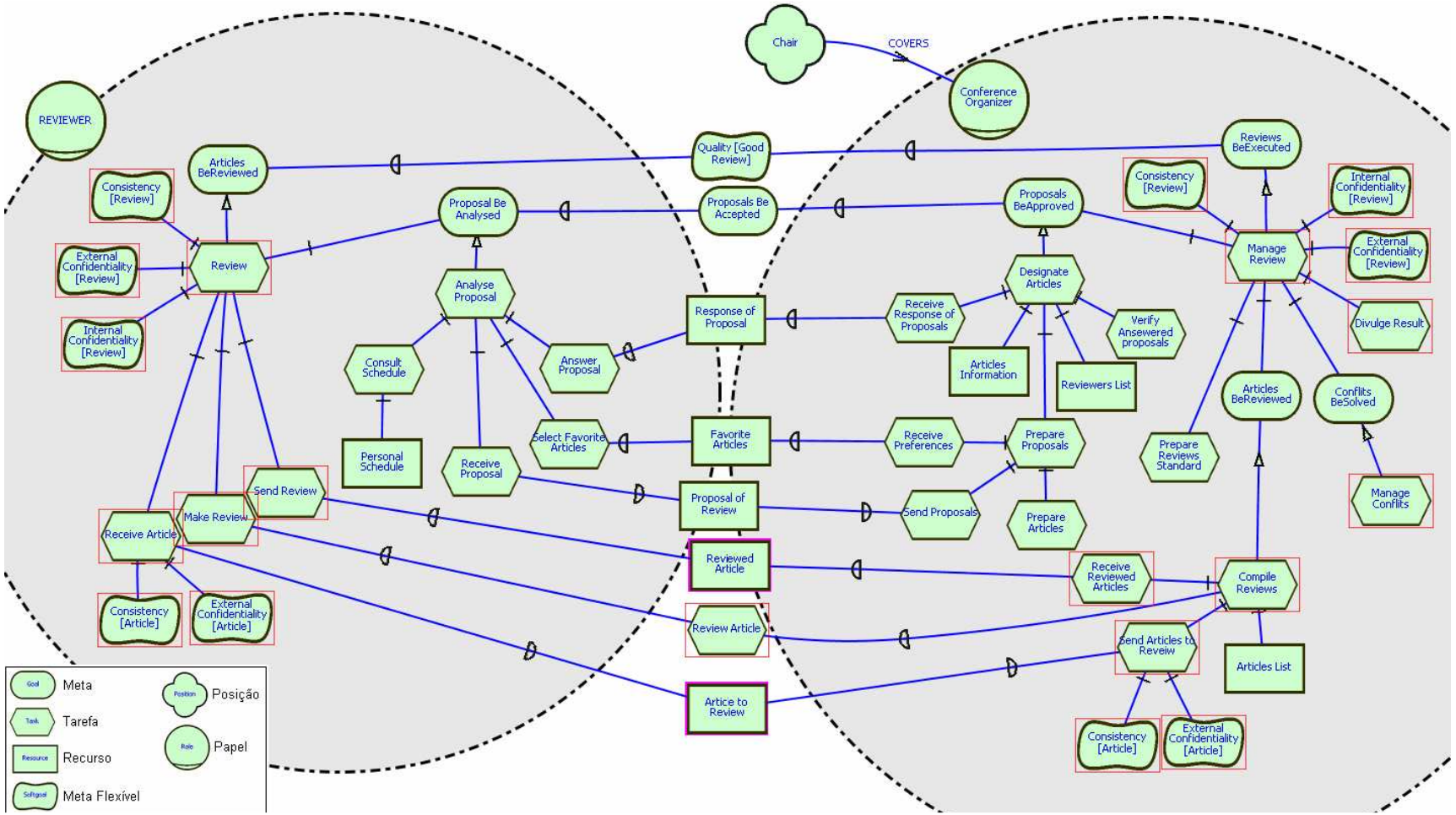

Figura 4.20 - Adição de metas flexíveis de segurança à situação de revisar artigo

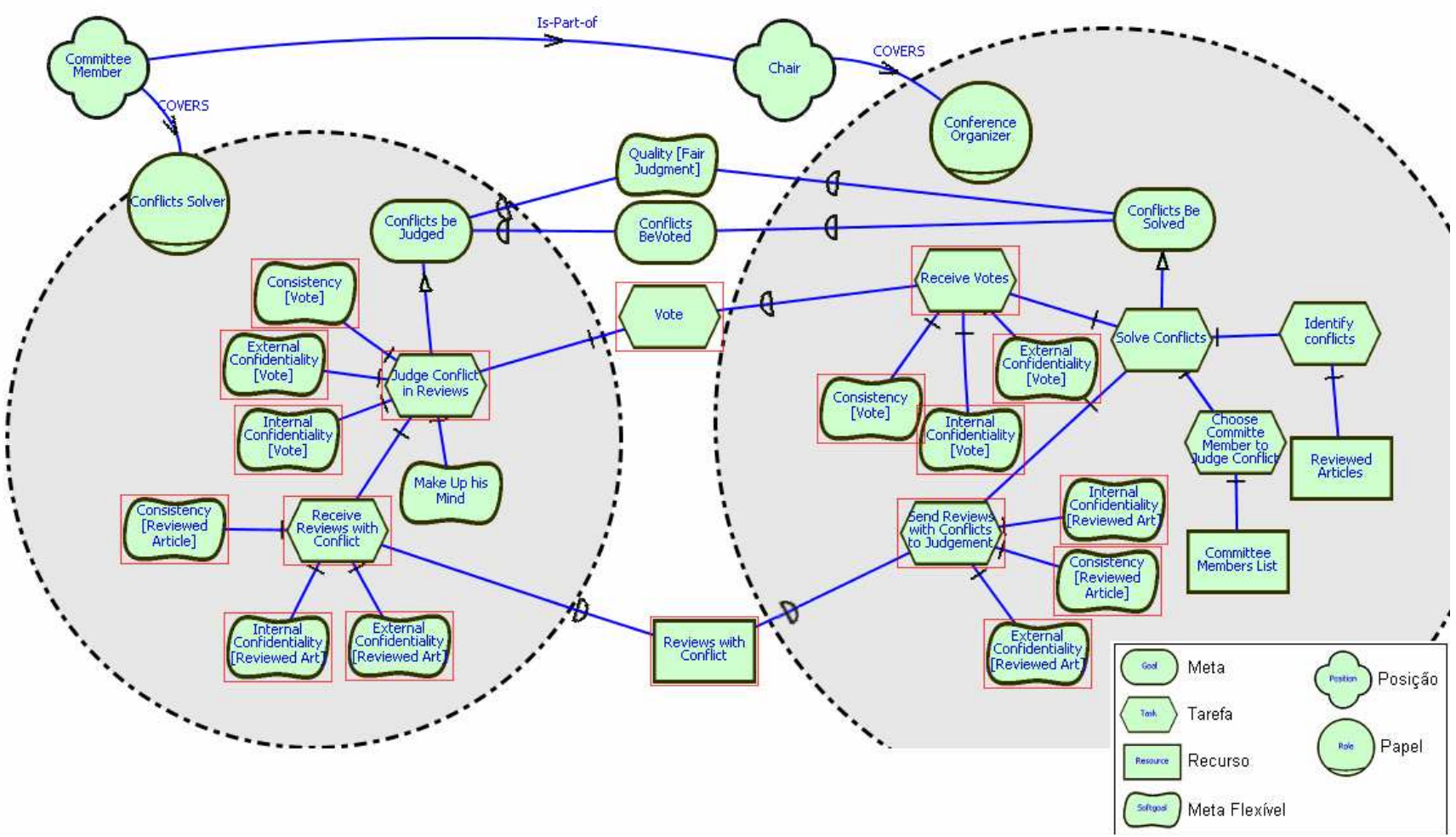

Figura 4.21 - Adição de metas flexíveis de segurança à situação de resolução de conflito

Passo 4 (Definição de requisitos específicos de segurança) – Identificação de medidas de ataque

Neste passo, são identificadas as medidas de ataque que os atacantes podem tomar para satisfazer suas intenções maliciosas. As medidas de ataque são modeladas como tarefas alternativas relacionadas com as intenções maliciosas, modeladas como metas, através de elos 'meios-fim'. Esta análise leva em consideração, além das intenções maliciosas dos atacantes, as vulnerabilidades do sistema que podem vir a ser exploradas.

A Figura 4.22, a seguir, apresenta medidas de ataque dos atacantes externos.

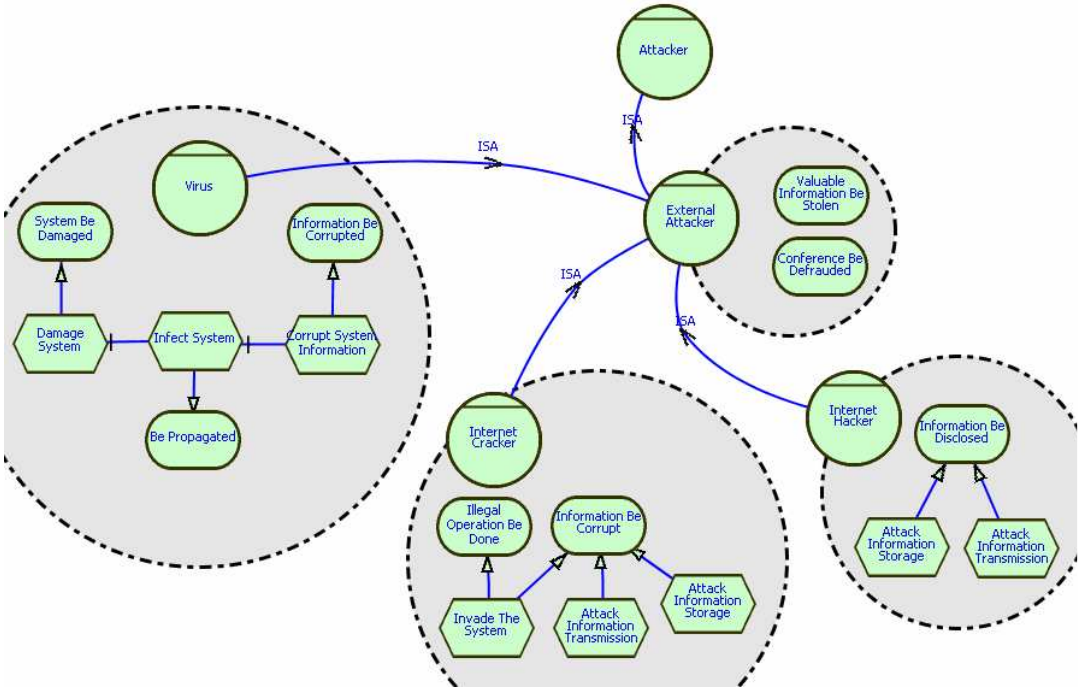

Figura 4.22 - Medidas de ataque dos atacantes externos.

Uma medida de ataque do agente Virus é 'danificar sistema', representada pela tarefa '*Damage System*'. Esta tarefa é uma alternativa para satisfazer sua intenção maliciosa 'sistema ser danificado', representada pela meta '*System Be Damaged*'. 'danificar sistema' é decomposta pela tarefa 'infectar sistema' (*Infect System*) que é ao mesmo tempo uma alternativa para satisfazer a intenção maliciosa de 'ser propagado', representada pela meta '*Be Propagated*'. 'Infectar sistema' é decomposta pela tarefa 'corromper informações do sistema' (*Corrupt System Information*) que é ao mesmo tempo uma alternativa para satisfazer a intenção

maliciosa de ‘informações ser corrompida’, representada pela meta ‘*Be Propagated’*. O agente ‘Internet Cracker’ tem como medidas de ataque ‘invadir o sistema’ (*Invade the System*), ‘atacar transmissão de informações’ (*Attack Information Transmission*) e ‘atacar armazenamento de informações’ (*Attack Information Storage*). ‘Invadir o sistema’ é uma alternativa para satisfazer ao mesmo tempo as intenções maliciosas de ‘operações ilegais serem realizadas’ e ‘informações serem corrompidas’. Já ‘atacar transmissão de informações’ e ‘atacar armazenamento de informações’ são alternativas apenas para a intenção maliciosa ‘informações serem corrompidas’. O agente ‘Internet Hacker’ tem como medidas de ataque para satisfazer sua intenção maliciosa ‘informações serem descobertas’ também as alternativas de ‘atacar transmissão de informações’ (*Attack Information Transmission*) e ‘atacar armazenamento de informações’ (*Attack Information Storage*).

Na Figura 4.23, a seguir, são apresentadas medidas de ataque dos atacantes internos. Como os atacantes são internos ao sistema, as medidas de ataque seguem a idéia que os atacantes tentarão se passar pelos atores legítimos, ou seja, tentarão desempenhar os papéis e ocupar as posições dos atores legítimos, se beneficiando de suas capacidades. Assim, para satisfazer suas intenções maliciosas o ‘atacante do Autor’ tem como tarefa alternativa ‘desempenhar o papel de Autor’ (*Plays Author Role*), que por sua vez é decomposta pela meta ‘personificação do agente legítimo pesquisador ser aceita’ (*Impersonation of Legitimate Researcher Be Accepted*). O ‘atacante do Revisor’ tem como tarefa alternativa ‘desempenhar o papel de Revisor’ (*Plays Review Role*), que por sua vez é decomposta pela meta ‘personificação do agente legítimo membro da equipe da conferência ser aceita’ - *Impersonation of Legitimate Conference Team Agent Be Accepted*. O ‘atacante do *Chair*’ tem como tarefa alternativa ‘ocupar a posição de *Chair*’ (*Occupies Chair Position*), que por sua vez é decomposta pela meta - *Impersonation of Legitimate Conference Team Agent Be Accepted*. O ‘atacante do Membro do comitê’ tem como tarefa alternativa ‘ocupar a posição de Membro do comitê’ (*Occupies Committee Member Position*), que por sua vez é decomposta pela meta ‘personificação do agente legítimo membro da equipe da conferência ser aceita’ - *Impersonation of Legitimate Conference Team Agent Be Accepted*. O ‘atacante do Coordenador’ tem como tarefa alternativa ‘desempenhar o papel de Coordenador’

(*Plays Review Role*), que por sua vez é decomposta pela meta ‘personificação do agente legítimo membro da equipe da conferência ser aceita’ - *Impersonation of Legitimate Conference Team Agent Be Accepted*.

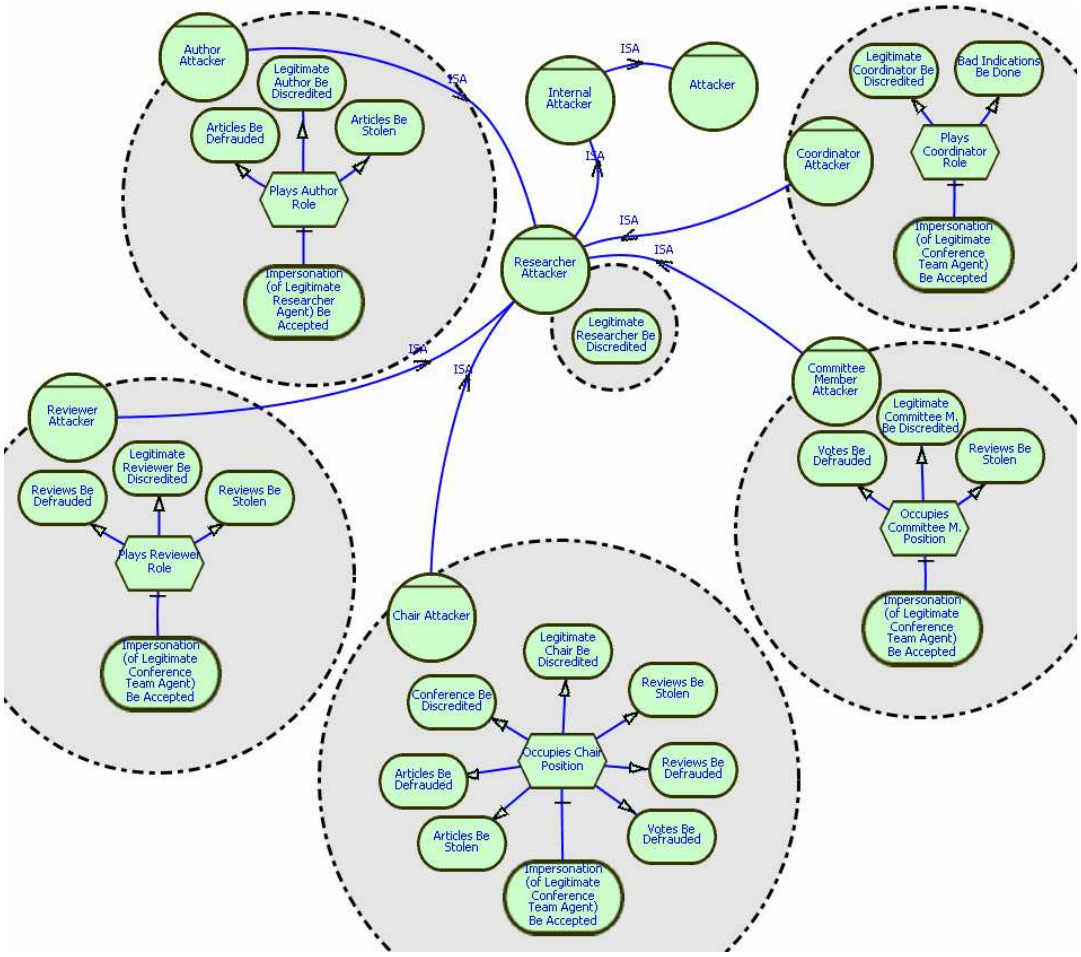

Figura 4.23 - Medidas de ataque dos atacantes internos.

Passo 5 (Definição de requisitos específicos de segurança) – Análise de medidas de defesa

O objetivo das medidas de defesa é ‘satisfazer a contento’ aos requisitos não-funcionais de segurança, definidos anteriormente no passo 10 (definição de requisitos gerais) – Definição de requisitos não-funcionais. As medidas de defesa são medidas capazes de conter as ameaças ao sistema (representadas pelas medidas de ataque identificadas no passo anterior). Estas medidas são as alternativas para operacionalização dos requisitos não-funcionais de segurança. Conforme mencionado no capítulo anterior, é importante destacar que algumas vulnerabilidades podem trazer danos a um sistema sem que haja necessariamente

um ataque. Isto pode ocorrer em caso de desastres naturais ou eventos aleatórios como falhas no funcionamento do *hardware*, por exemplo. A análise das medidas de defesa deve considerar também estes outros eventos.

Operacionalização dos requisitos não-funcionais

Em geral, um requisito não-funcional pode ser implementado de mais de uma forma. Este passo inicia-se com a elicitação das alternativas para operacionalização dos requisitos não-funcionais, previamente definidos no passo 10 (definição de requisitos gerais). As alternativas para operacionalização dos requisitos não-funcionais também podem ser elicitadas a partir de catálogo [7] [20] e instanciadas para alternativas de operacionalização específicas da aplicação. Novas alternativas de operacionalização também poder ser propostas e posteriormente adicionadas aos catálogos de requisitos para reuso futuro, enriquecendo assim o catálogo de requisitos não funcionais. Além dos catálogos de requisitos, alternativas de operacionalização podem ser elicitadas a partir de normas e guias de boas práticas de segurança, como [22] e [26].

Uma determinada alternativa de operacionalização de um requisito não funcional pode ter impacto, e geralmente tem, em outros requisitos não-funcionais. A escolha da melhor alternativa para operacionalização deve ser feita de acordo com a prioridade dos requisitos impactados.

A Figura 4.24, a seguir, apresenta algumas alternativas para operacionalização dos requisitos de confidencialidade e consistência.

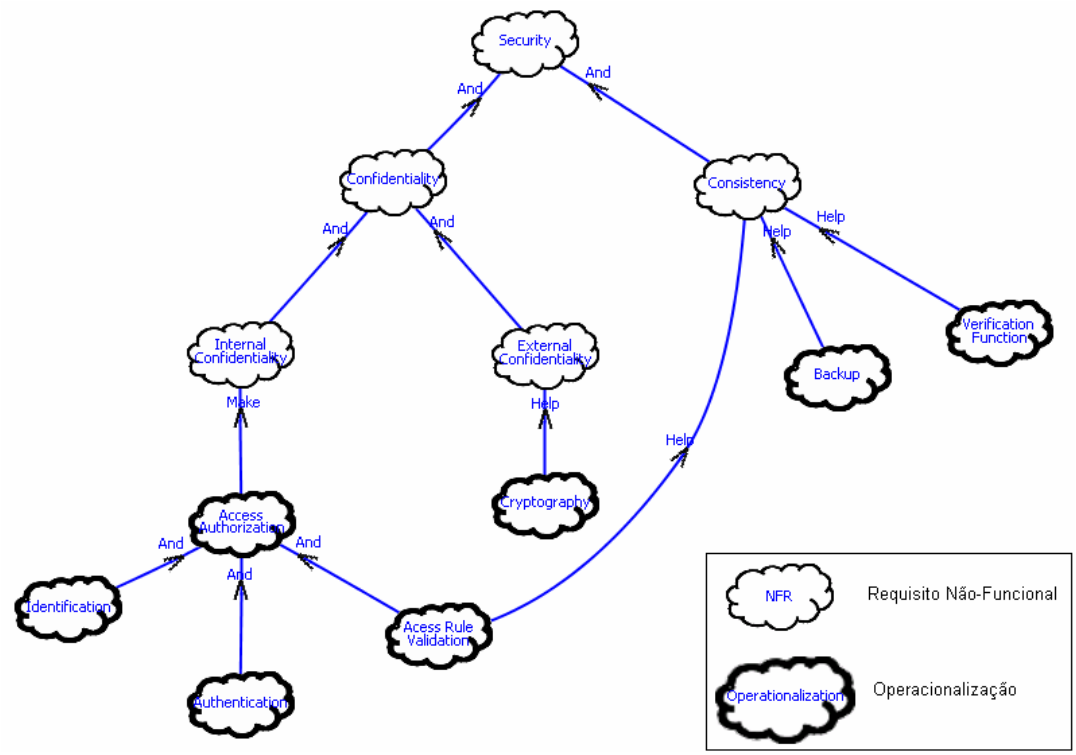

Figura 4.24 - Alternativas de operacionalização para confidencialidade e consistência.

Confidencialidade Interna

Na análise de medidas de defesa para o requisito não-funcional ‘confidencialidade interna [artigo revisado]’ (*Internal Confidentiality [Reviewed Article]*), apresentada na Figura 4.25, a alternativa para operacionalização é representada pela tarefa ‘autorizar acesso’ (*Autorize Access*) que é decomposta pelas sub-metas ‘acesso ser validado’ (*Access Be Validated*), ‘usuário ser identificado’ (*User Be Identified*) e ‘usuário ser autenticado’ (*User Be Autheticated*). As sub-metas ‘usuário ser identificado’ e ‘usuário ser autenticado’ têm o objetivo de impedir que atacantes consigam se passar por atores legítimos e a sub-meta ‘acesso ser validado’ tem o objetivo de controlar quais atores tem autorização para acessar as informações, neste caso, o recurso artigo revisado.

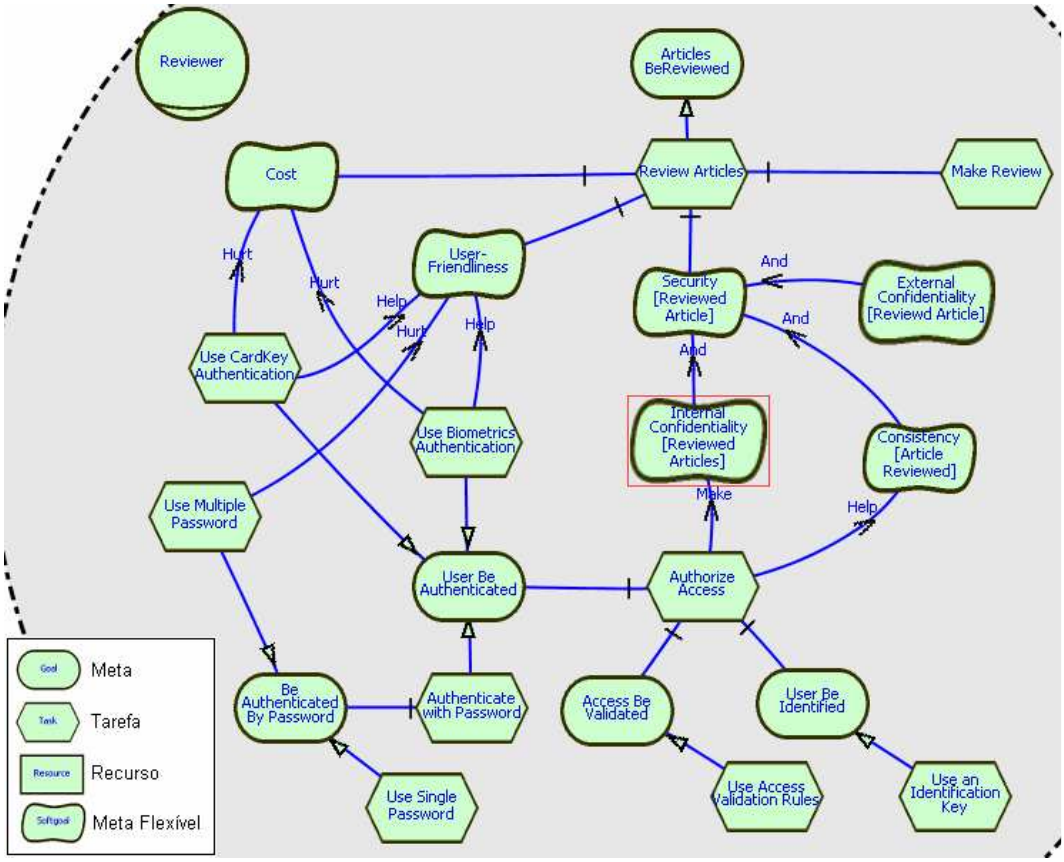

Figura 4.25 - Análise de medidas de defesa - confidenciadade interna [artigo revisado]

Alternativas:

Para satisfazer a meta ‘usuário ser autenticado’ há três alternativas: ‘usar autenticação com cartão’ (*Use CardKey Authentication*), ‘usar autenticação biométrica’ (*Use Biometrics Authentication*) e ‘autenticar com senha’ (*Authenticate with Password*).

A tarefa ‘autenticar com senha’ é decomposta pela meta ‘ser autenticado por senha’ (*Be Authenticated By Password*) que possui duas alternativas: ‘usar senha simples’ (*Use Single Passoword*) e ‘usar senhas múltiplas’ (*Use Multiple Password*). Para satisfazer a sub-meta ‘acesso ser validado’ a alternativa é a tarefa ‘usar regras de validação de acesso’ (*Use Access Validation Rules*) e para satisfazer a sub-meta ‘usuário ser identificado’ a alternativa é a tarefa ‘usar uma chave de identificação’ (*Use an Identification Key*).

Análise:

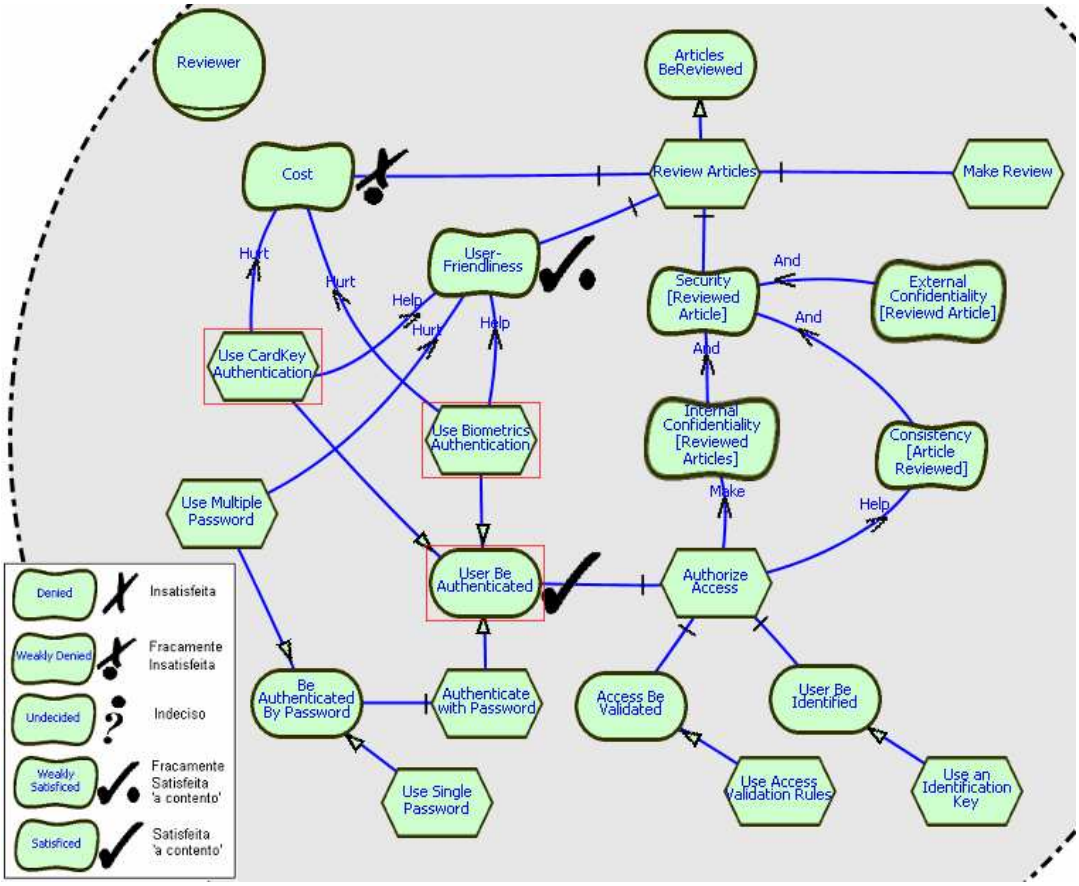

Figura 4.26 - Análise de alternativas: ‘usar autenticação com cartão’ e ‘usar autenticação biométrica’

A alternativa ‘usar senhas múltiplas’, apresentada na Figura 4.27, também fere a usabilidade (uma vez que o usuário é obrigado a decorar mais de uma senha), sendo neutra em relação ao custo. Tanto a alternativa ‘usar senhas múltiplas’ quanto a alternativa ‘usar senha simples’ satisfazem a meta ‘ser autenticado por senha’. Uma vez satisfeita esta meta, a meta ‘usuário ser autenticado’ também é satisfeita, devido à propagação do rótulo de contribuição [7] (da meta ‘ser autenticado por senha’ para a meta ‘usuário ser autenticado’). Neste exemplo, e nos próximos, a propagação dos rótulos foi feita de forma manual.

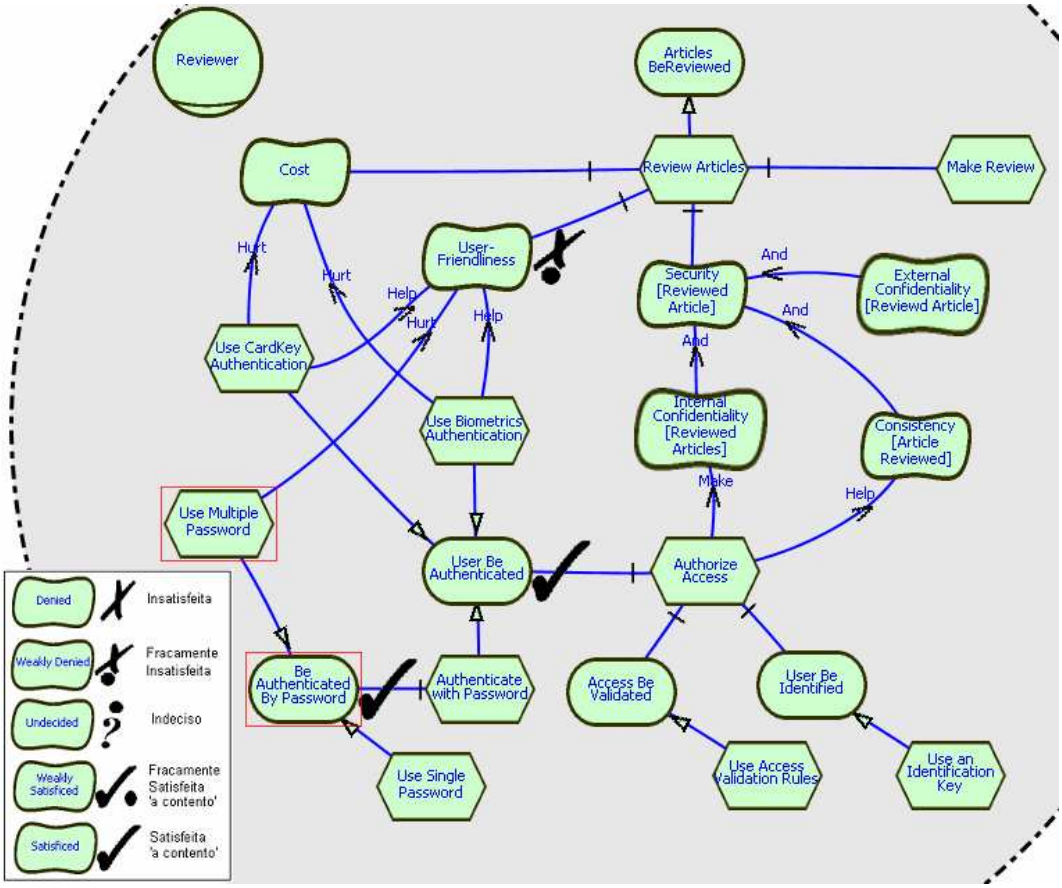

Figura 4.27 - Análise de alternativas: ‘usar senhas múltiplas’.

A tarefa ‘usar senha simples’ é considerada neutra em relação aos requisitos de custo e usabilidade, conforme apresentado na Figura 4.28.

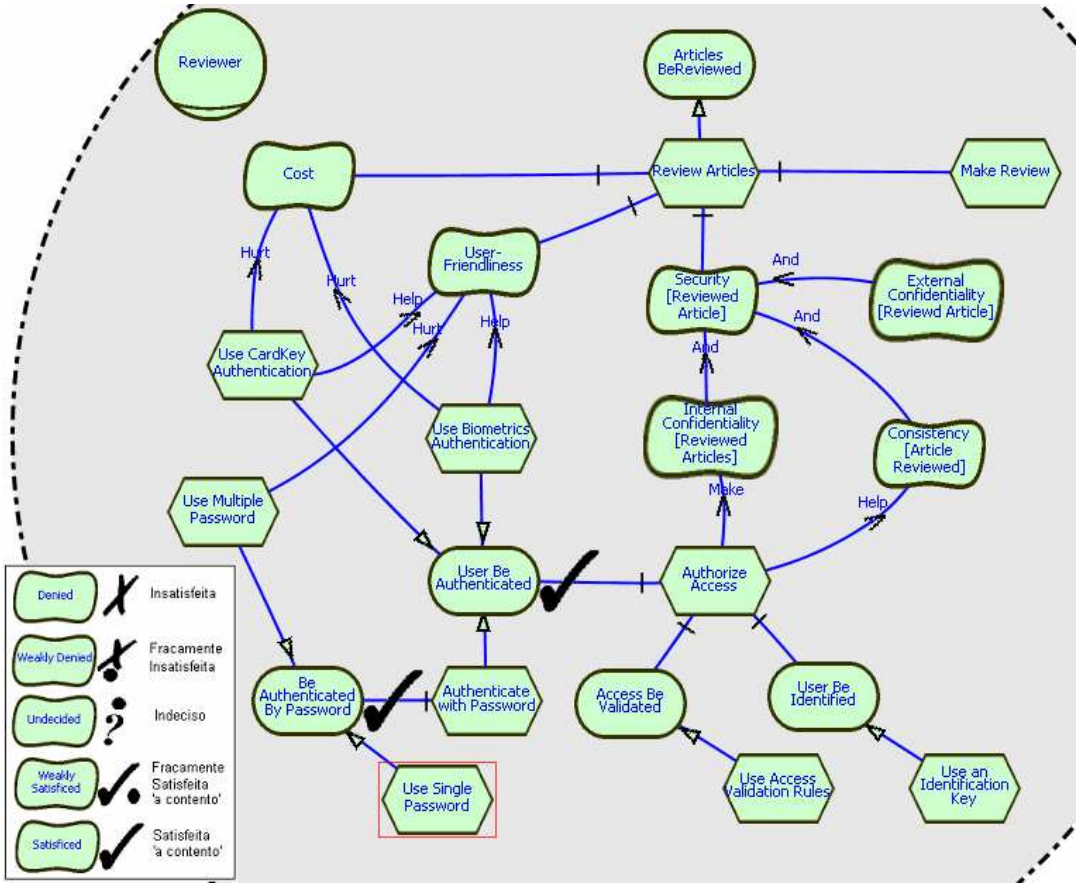

Figura 4.28 - Análise de alternativas: ‘usar senha simples’

Escolha da alternativa:

A escolha da alternativa mais adequada depende da prioridade dada aos requisitos não-funcionais de custo e usabilidade além de segurança. Em um cenário onde custo e usabilidade tenham a mesma prioridade, a escolha da alternativa ‘usar senha simples’ se mostra adequada. Justamente com as alternativas ‘usar regras de validação de acesso’ e ‘usar uma chave de identificação’, também consideradas neutras em relação ao custo e usabilidade, estas alternativas ‘satisfazem a contento’ a meta flexível de ‘confidencialidade interna [Artigo Revisado]’. Ao mesmo tempo, estas alternativas impedem (break) que a meta ‘personificação do agente legítimo membro da equipe da conferência ser aceita’ (do agente Atacante do Revisor) seja satisfeita, conforme apresentado na Figura 4.29.

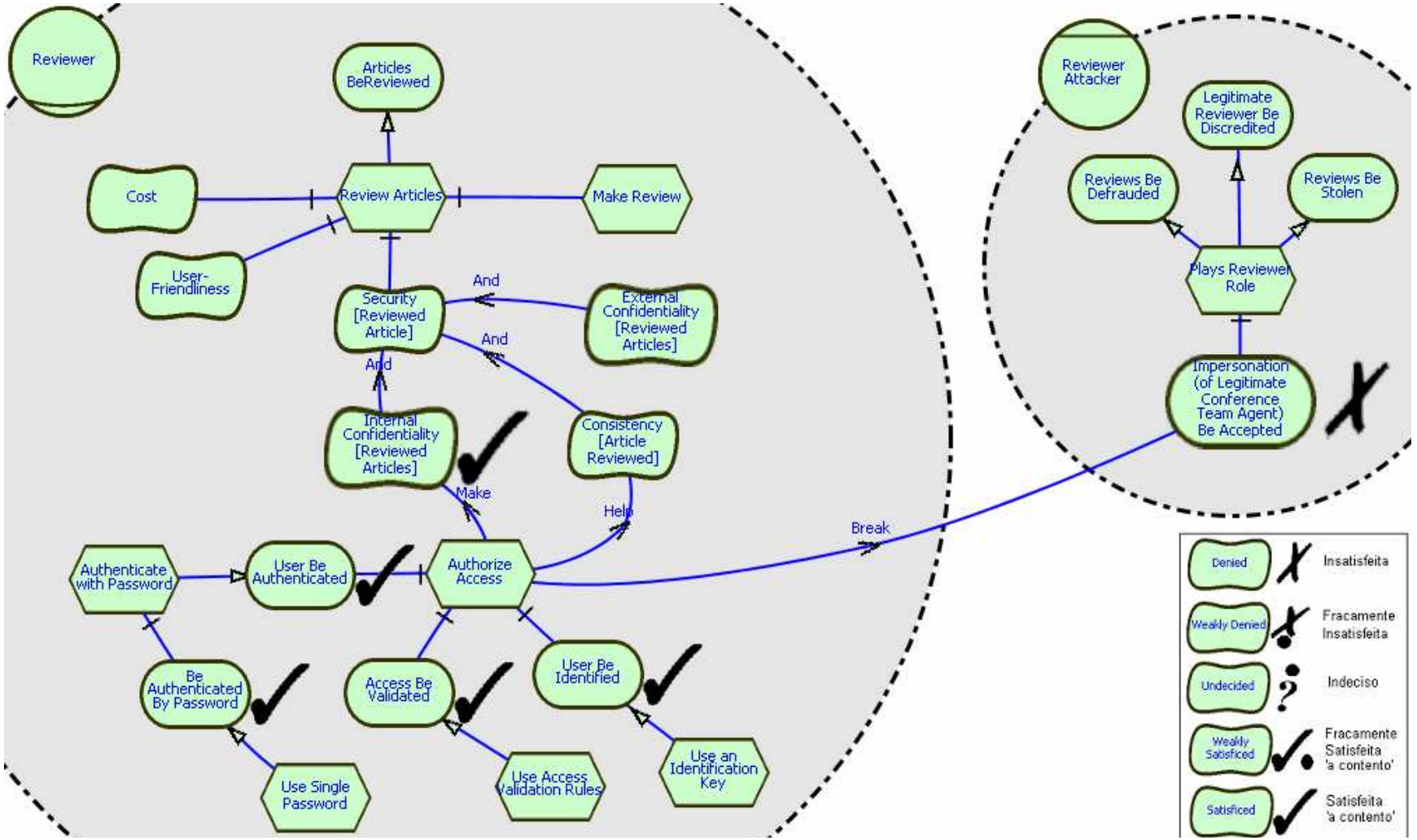

Figura 4.29 - Escolha da alternativa: ‘usar senha simples’

Um mecanismo auxiliar para a escolha de alternativas é a análise de variabilidade visual proposta em [35] para modelos orientados a metas. Feitas as correlações entre alternativas de operacionalização e requisitos não-funcionais (via elos de contribuição) é possível, através deste mecanismo, analisar visualmente (em forma de gráficos) o impacto, positivo ou negativo, que a escolha de um determinado conjunto de alternativas de operacionalização tem na satisfação dos requisitos não-funcionais.

Confidencialidade Externa

A análise de medidas de defesa para ‘satisfazer a contento’ o requisito de confidencialidade externa do artigo revisado, conforme Figura 4.30, indica a tarefa ‘criptografar’ (*Encrypt*) como alternativa para operacionalização deste requisito não-funcional e esta tarefa é decomposta pela sub-meta ‘informação ser criptografada’ (*Information Be Encrypted*). Através desta alternativa, a confidencialidade fica protegida de ataques externos, uma vez que as informações estarão criptografadas, o que impede a descoberta do seu real conteúdo.

Alternativas:

Para satisfazer a meta ‘informação ser criptografada’ são apresentadas duas alternativas: ‘usar chave de 64 bits’ (*Use 64bits Key*) e ‘usar chave de 128bits’ (*Use 128bits Key*).

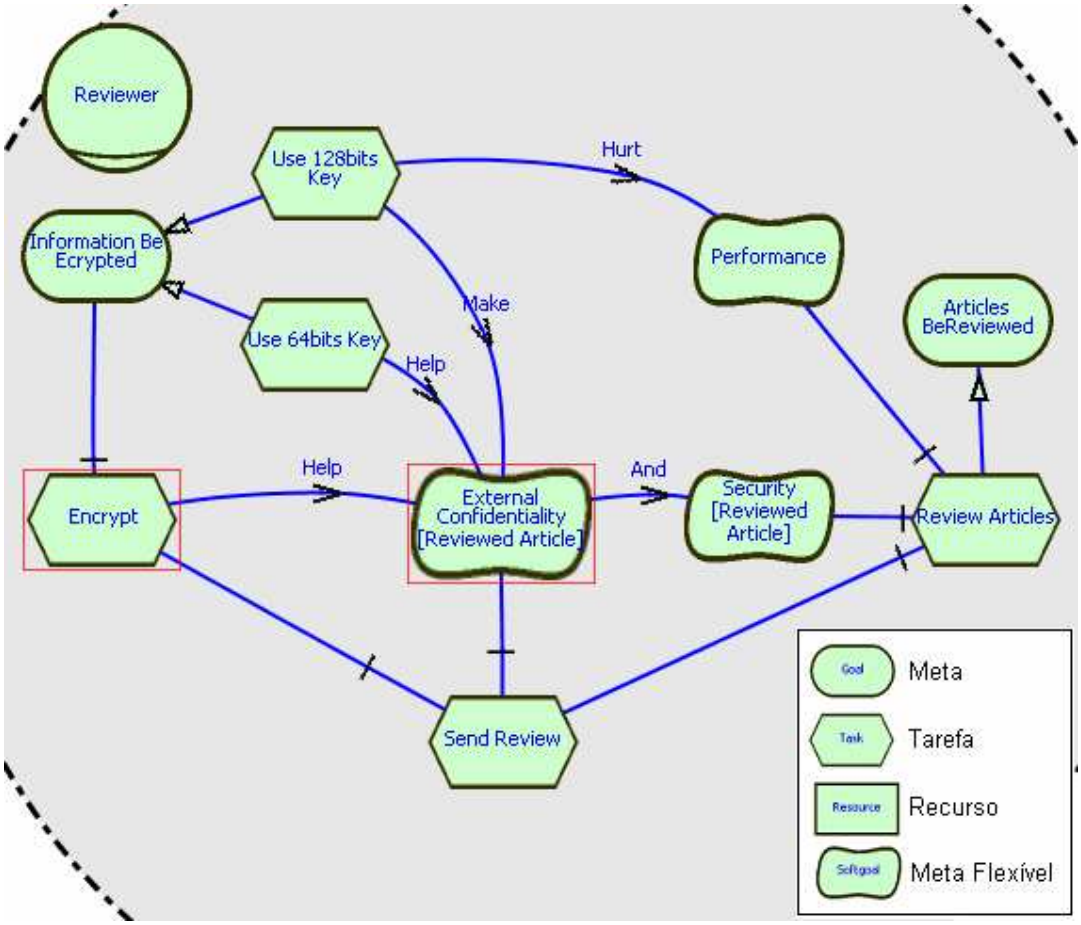

Figura 4.30 - Análise de medidas de defesa para confidencialidade externa (artigo revisado)

Análise:

A alternativa ‘usar chave de 64bits’ ajuda (contribui positivamente para) o requisito não-funcional de ‘confidencialidade externa’, e é considerada neutra em relação ao requisito não-funcional de desempenho (*Performance*), conforme apresentado na Figura 4.31.

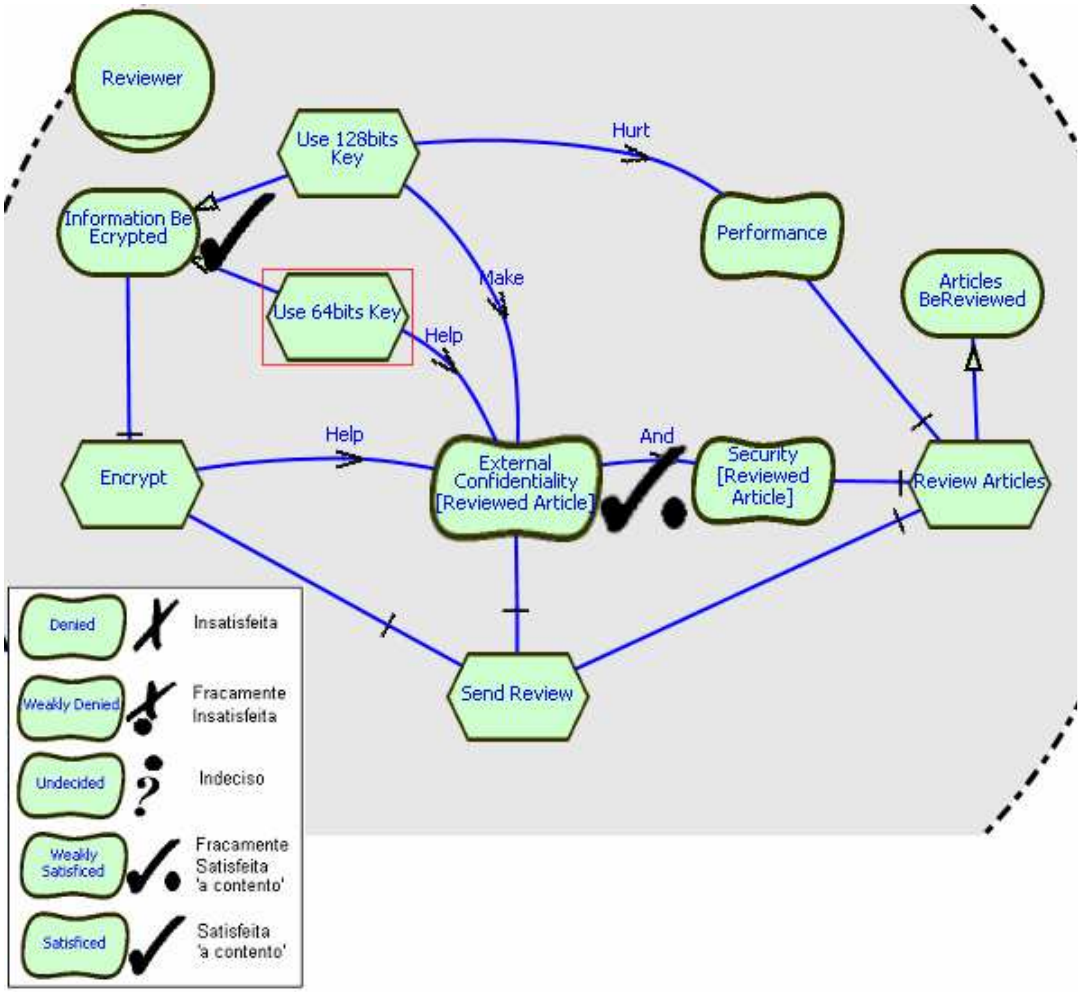

Figura 4.31 - Análise de alternativas: ‘usar chave de 64bits’

A alternativa ‘usar chave de 128bits’ garante a ‘satisfação a contento’ do requisito não-funcional de ‘confidencialidade externa’, mas fere (contribui negativamente para) o requisito não-funcional de desempenho (*Performance*), conforme apresentado na Figura 4.32.

Escolha da alternativa:

A escolha da alternativa mais adequada depende da prioridade dada aos requisitos não-funcionais impactados, neste caso, desempenho e confidencialidade externa.

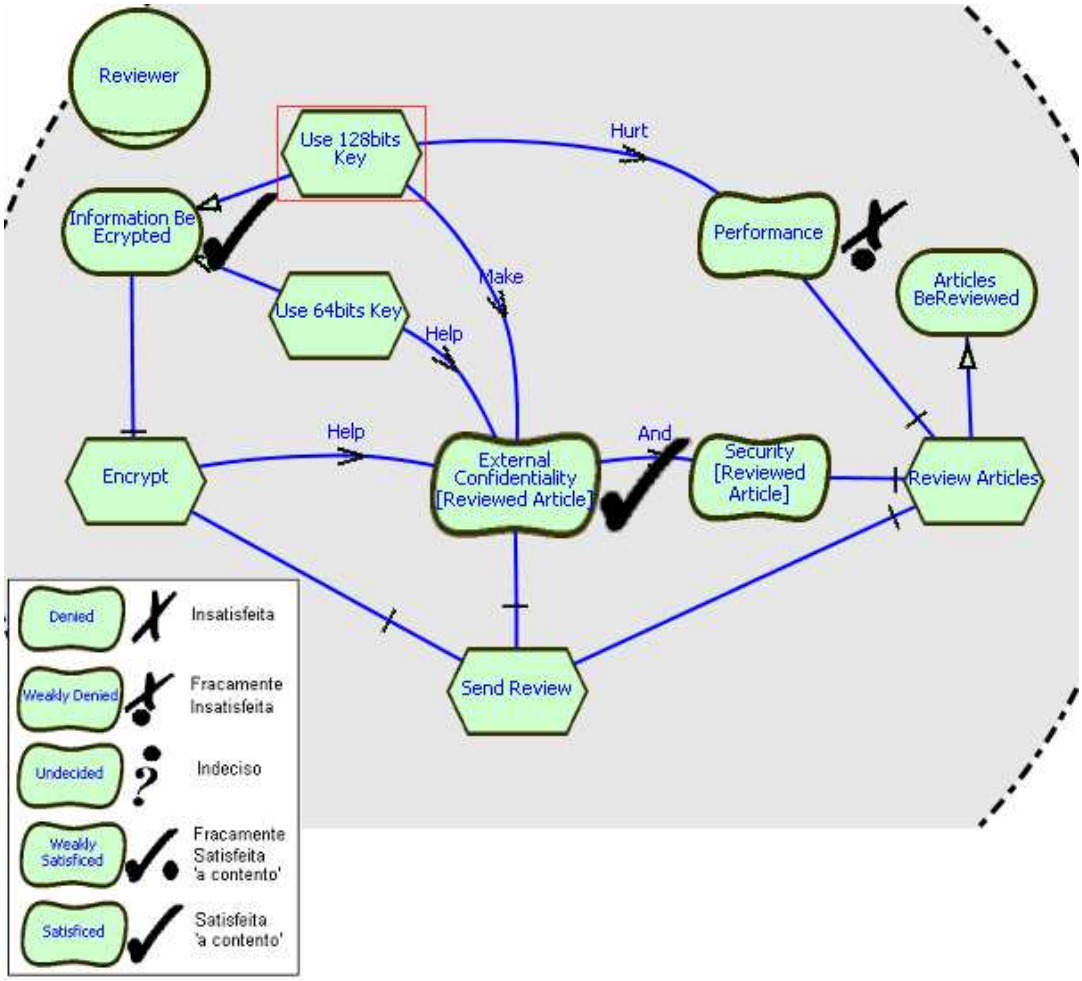

Figura 4.32 - Análise de alternativas: ‘usar chave de 128bits’

Em um cenário onde a confidencialidade externa tenha prioridade em relação ao desempenho, a escolha da alternativa ‘usar chave de 128bits’ é mais adequada, conforme Figura 4.33. Esta alternativa ‘satisfaz a contento’ o requisito de confidencialidade externa [artigo revisado] ao mesmo tempo que impede (*break*) a satisfação da meta ‘informação ser descoberta’ do agente Internet *Hacker*.

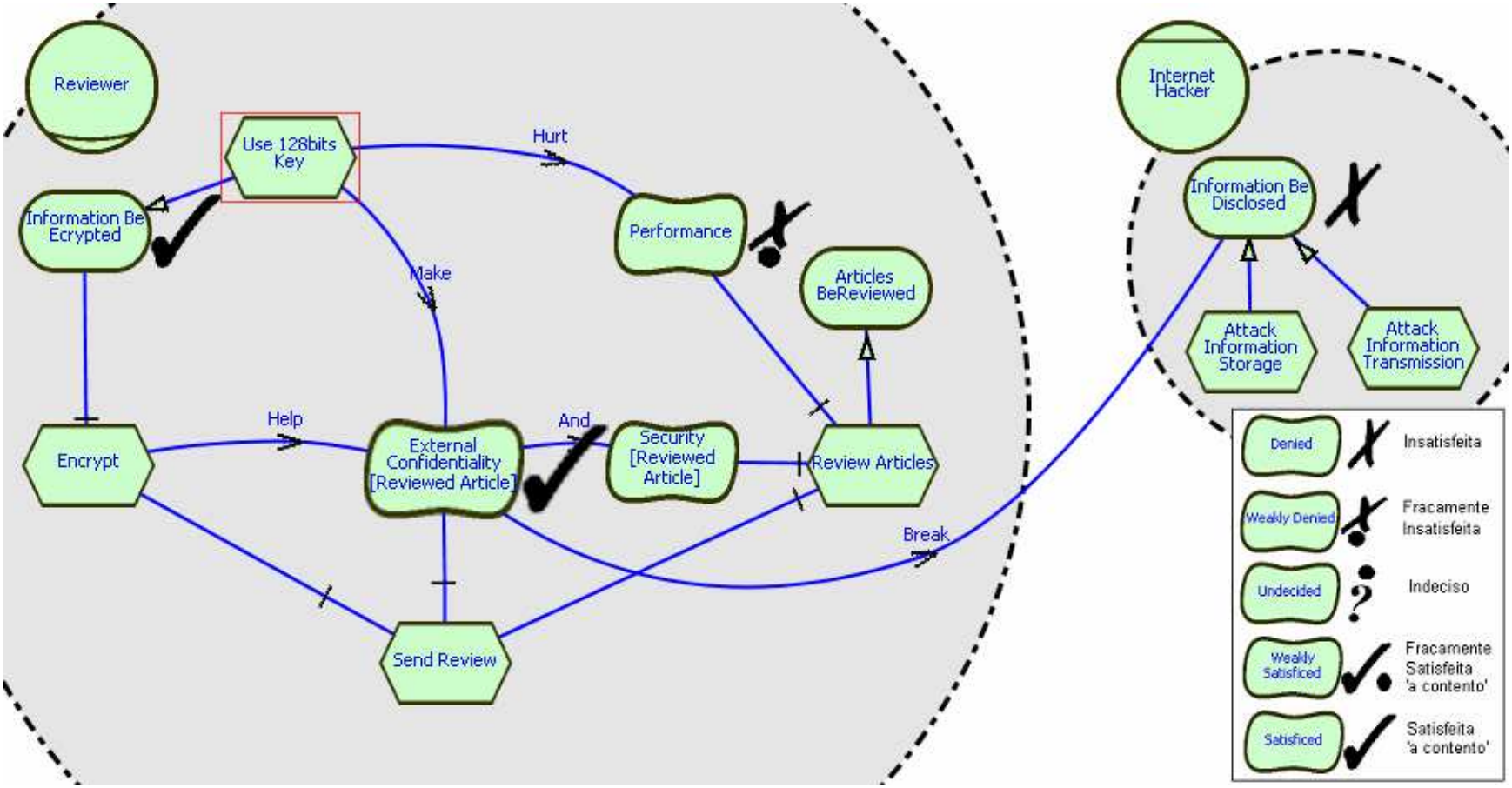

Figura 4.33 - Escolha da alternativa: ‘usar chave de 128bits

É importante ressaltar que a alternativa escolhida para operacionalização de confidencialidade externa deve ser implementada não apenas na transmissão das informações (em virtude de um possível 'ataque a transmissão de informações'), mas também no armazenamento das informações (em virtude de um possível 'ataque ao armazenamento de informações').

Consistência

De acordo com a análise de medidas de defesa para 'satisfazer a contento' o requisito de consistência (artigo revisado), conforme Figura 4.34, são necessárias duas tarefas: 'autorizar acesso' (*Authorize Access*) e 'checar consistência' (*Check Consistency*). A tarefa 'autorizar acesso' tem o objetivo de permitir que a informação seja alterada apenas pelos atores autorizados, de acordo com as regras de validação de acesso. A decomposição da tarefa 'autorizar acesso' e suas alternativas de operacionalização foram discutidas anteriormente na análise de medidas de defesa para o requisito de confidencialidade interna. A tarefa 'checar consistência' é decomposta em duas sub-metas: 'consistência da informação ser checada' (*Information Consistency Be Checked*) e 'informação ser copiada' (*Information Be Copied*). A sub-meta 'consistência da informação ser checada' tem o objetivo de verificar se a informação não foi corrompida, enquanto a sub-meta 'informação ser copiada' tem o objetivo de guarda uma cópia de segurança, possibilitando a restauração da mesma caso seja necessário. A tarefa 'checar consistência', com sua respectiva decomposição, é uma medida de defesa também para casos de desastre natural ou falhas de equipamento, onde não há ataque.

Alternativas:

Para operacionalização da sub-meta 'consistência da informação ser checada' a alternativa é a tarefa 'usar função *hash*' (*Use Hash Function*). Este tipo de função gera um código de tamanho pequeno que serve como 'impressão digital' para determinada informação. Se a informação for alterada, o código gerado será diferente do original e será detectado que houve uma alteração na informação.

Para operacionalização da sub-meta ‘informação ser copiada’ há duas alternativas: a tarefa ‘usar mídia de DVD’ e a tarefa ‘usar mídia de disco magnético’.

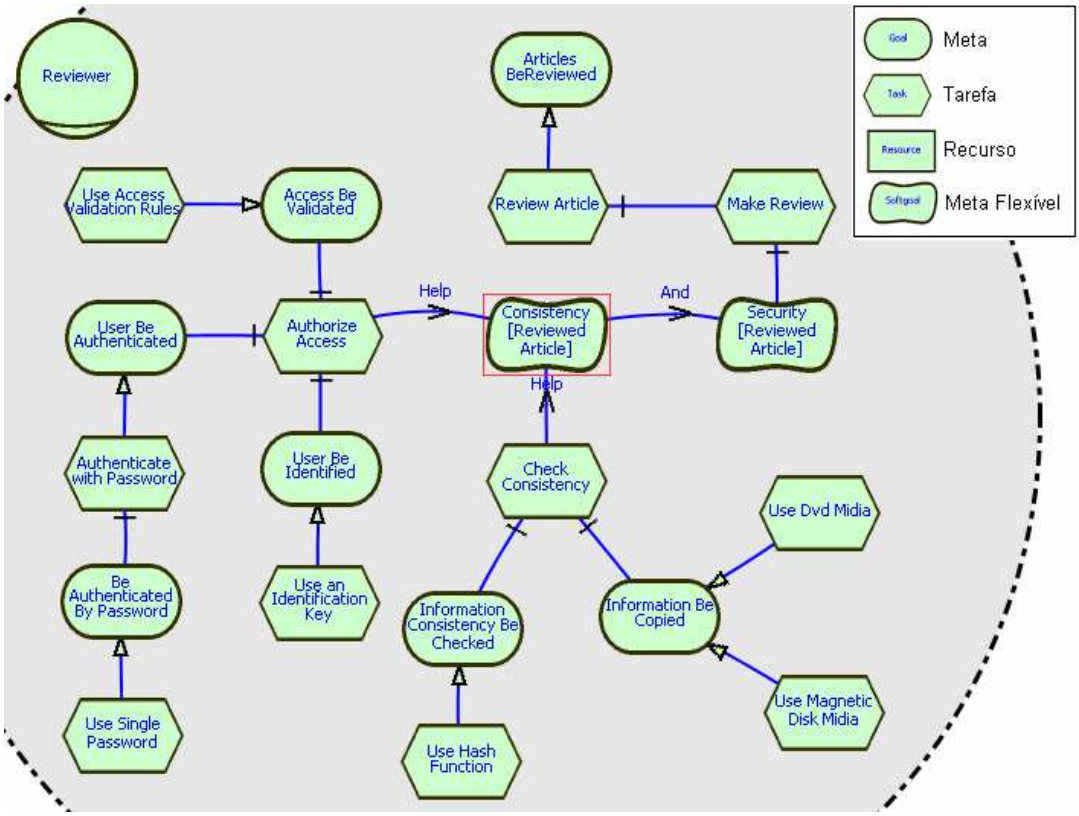

Figura 4.34 - Análise de medidas de defesa para consistência (artigo revisado).

Escolha da alternativa:

Tanto a alternativa ‘usar mídia de DVD’ quanto a alternativa ‘usar mídia de disco magnético’ satisfazem a meta ‘informação ser copiada’ e não ferem nenhum outro requisito não-funcional. Neste caso, a escolha de qualquer das duas é válida. A Figura 4.35 apresenta a escolha da alternativa ‘usar mídia de DVD’ e a alternativa de ‘usar função *hash*’ para operacionalização da sub-meta ‘informação ser checada’. Estas tarefas alternativas juntamente com a tarefa ‘autorizar acesso’ conseguem ‘satisfazer a contento’ ao requisito não-funcional de consistência, ao mesmo tempo que impedem (*break*) a satisfação da meta ‘informação ser corrompida’ dos agentes Vírus e Internet *Cracker*.

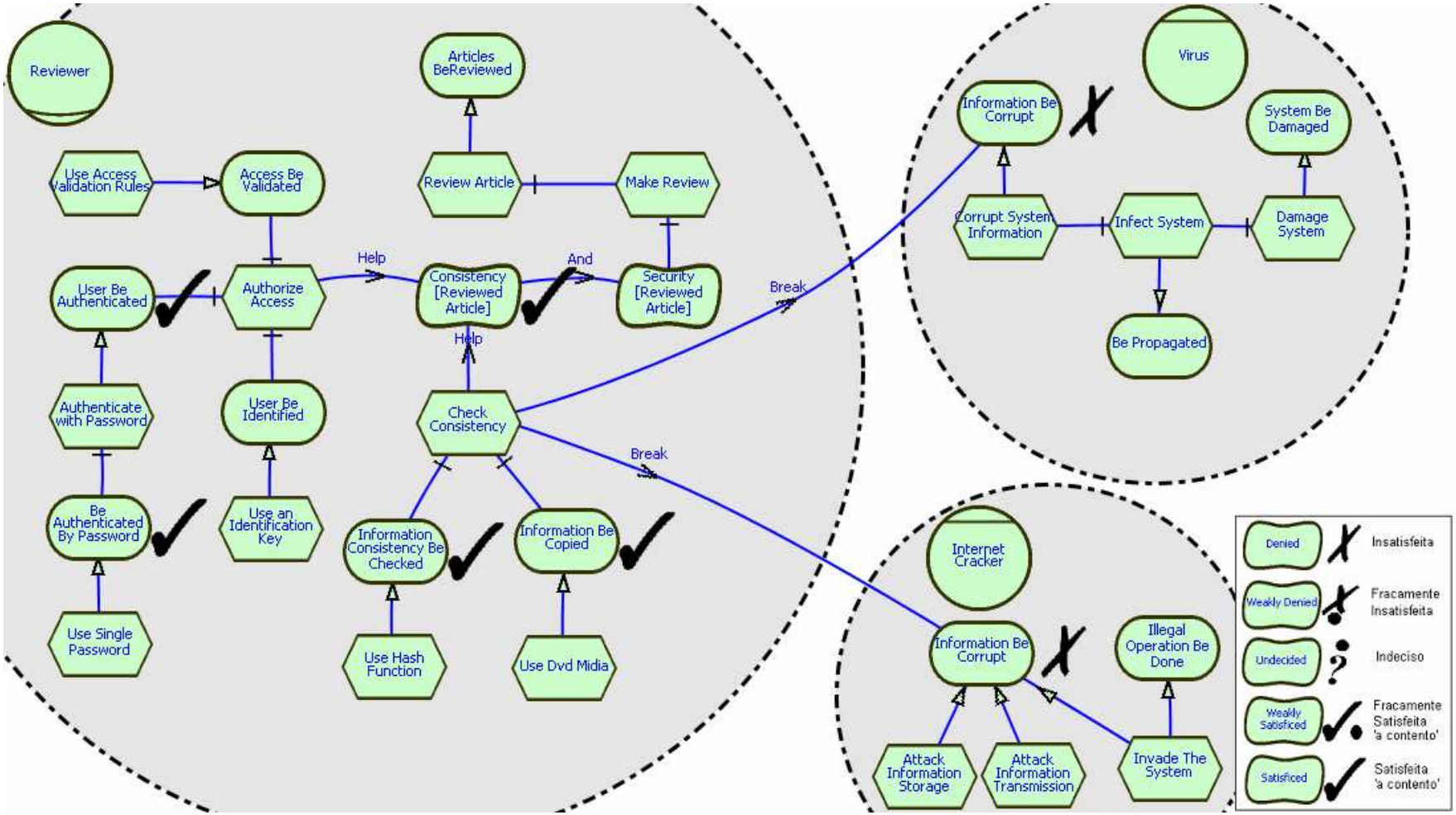

Figura 4.35 - Escolha de alternativa: ‘usar mídia de DVD’

A operacionalização destes requisitos não-funcionais pode ser reusada como medida de defesa para as vulnerabilidades identificadas nas demais situações. A Figura 4.36, a seguir, apresenta o reuso da operacionalização dos requisitos de confidencialidade externa e consistência na situação de submissão de artigo.

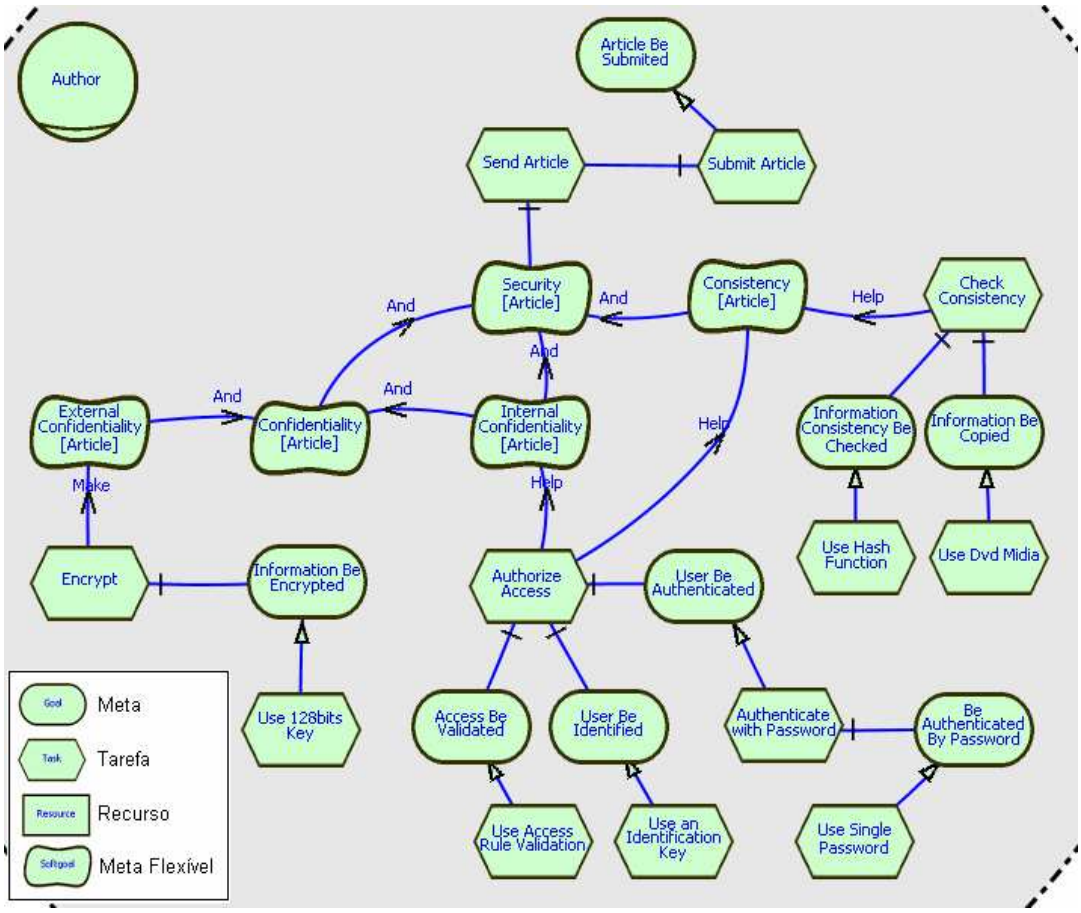

Figura 4.36 - Operacionalização de confidencialidade interna e externa e consistência na 'submissão de artigo'

A Figura 4.37, a seguir, apresenta o reuso da operacionalização dos requisitos de confidencialidade interna e externa e consistência na situação de 'resolução de conflito'.

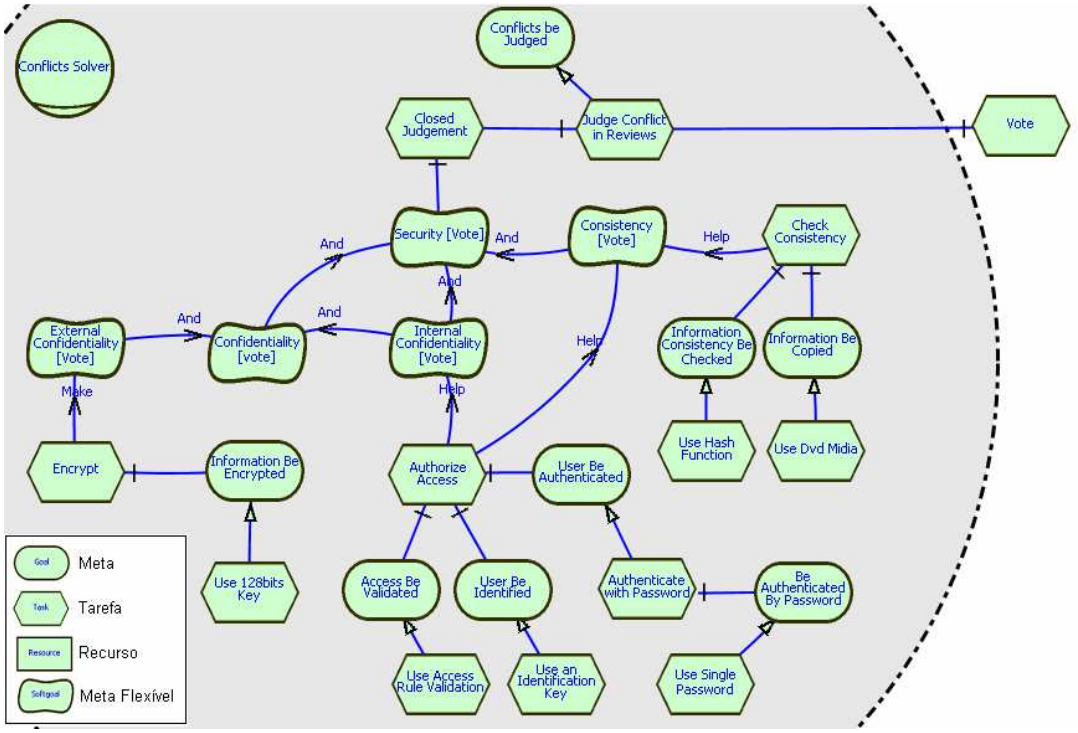

Figura 4.37 - Operacionalização de confidencialidade interna e externa e consistência na 'resolução de conflito'

Uma alternativa para a situação de 'resolução de conflito' é ter a votação aberta. Com a votação aberta abra-se mão do requisito de confidencialidade interna, uma vez que os Membros do comitê tomariam conhecimento dos votos uns dos outros. Por outro lado, esta opção simplifica consideravelmente a operacionalização da segurança do voto. Em uma votação aberta, o registro das opiniões dos Membros do comitê pode ser feito fisicamente em uma ata, e para 'satisfazer a contento' os requisitos de consistência e confidencialidade da opinião (voto) dos Membros do comitê é necessário restringir fisicamente o acesso a esta ata, conforme apresentado na Figura 4.38 a seguir.

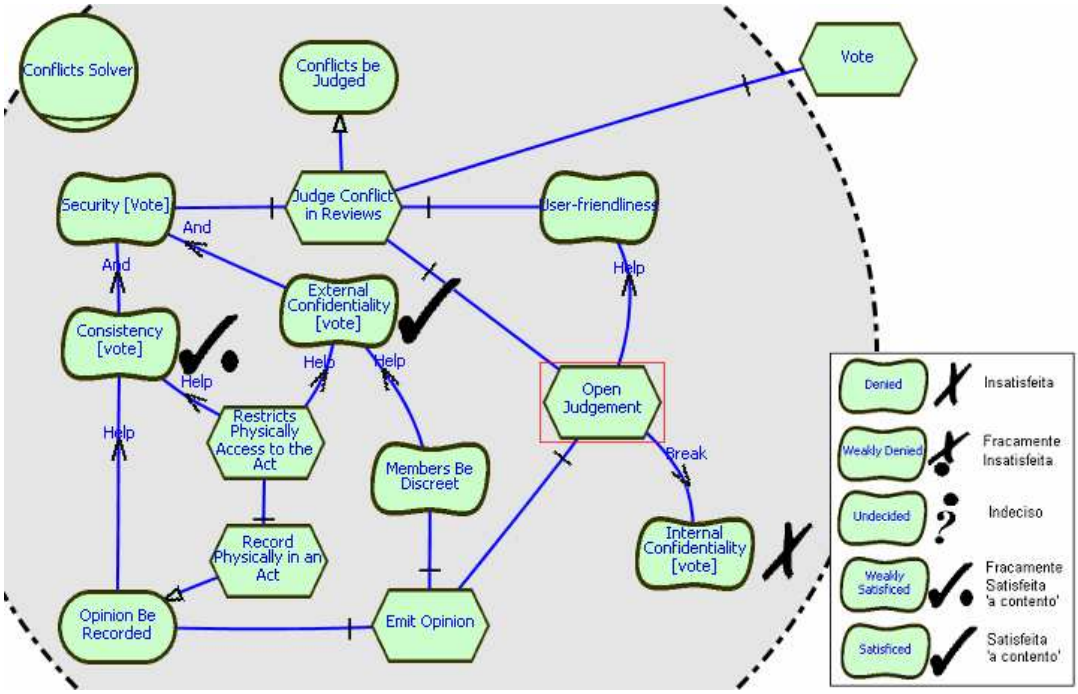

Figura 4.38 - Votação aberta na 'resolução de conflito'

Passo 6 - iterativo (Definição de requisitos específicos de segurança) – Identificação de novas vulnerabilidades

Neste passo, deve-se checar se novas vulnerabilidades foram introduzidas em função das alternativas de defesa escolhidas. Como as alternativas de defesa tratadas até aqui não produziram novos recursos não tratados, não foram introduzidas novas vulnerabilidades em relação à confidencialidade ou à consistência de informações.

Passo 7 - (Definição de requisitos específicos de segurança) – Incorporação das medidas de defesa escolhidas

As medidas de defesa escolhidas são incorporadas à definição de requisitos gerais. Com isto, os cenários correspondentes são refinados, incorporando a opção escolhida para operacionalização dos requisitos não-funcionais. Os cenários refinados são apresentados a seguir. As linhas referentes aos requisitos não-funcionais estão em azul e as linhas referentes à sua operacionalização estão em vermelho.

Scenario definition		
Title:	**Article Submission**	
Goal:	Article be submitted.	
Context:		
Geographical Location:	WEB	
Temporal Location:	Before the submission deadline.	
Constraint:		
Precondition:		
Resources:	Computer, Internet, Article	
Constraint:		
Actors:	Chair, Author	
Episodes:	Chair informs the event dates	
	Author prepare the article.	
	Author sends the article to Chair.	
	Consistency [Article]	
	Check Consistency of Article	
	Use hash function to generate an article's 'fingerprint'	
	External Confidentiality [Article]	
	Encrypt Article	
	Use a 128bits Key (to encrypt article)	
	Chair receives article	
	Chair receives article by System	
	External Confidentiality [Article]	
	Decrypt Article	
	Use a 128bits Key (to decrypt article)	
	Consistency [Article]	
	Check Consistency of Article	
	Use DVD midia to make an Article's copy	
	Chair updates the Articles List	
Constraint:	NFR: Security: *Consistency [Article]*, *External Confidentiality [Article]*; User-friendliness	
Exceptions:		

Scenario definition		
Title:	**Articles Review**	
Goal:	ArticlesBeReviewed	
Context:		
Geographical Location:	WEB	
Temporal Location:	Before deadline to send reviews.	
Constraint:		
Precondition:	Proposal Acceptance.	
Resources:	Computer, Internet, Articles to review	
Constraint:		
Actors:	Chair, Reviewers	
Episodes:	Chair prepares standards to review	
	Chair selects an article to sends to review	
	Chair selects reviewers that have accepted the proposal of review for the selected article;	
	Chair sends the article to each reviewer selected (in previous step)	
	External Confidentiality [Article]	
	Encrypt Article	
	Use a 128bits Key (to encrypt article)	
	Reviewer receives the article sent by Chair	
	External Confidentiality [Article]	
	Decrypt Article	
	Use a 128bits Key (to decrypt article)	
	Reviewer makes the article review	
	Internal Confidentiality [Reviewed Article]	
	Authorize Access	
	Reviewer uses an Identification Key	
	Authenticate Reviewer with Password	
	Reviewer uses Single Passoword	
	Uses Access Validation Rules to check if Reviewer can make the review to this article	
	Consistency [Reviewed Article]	
	Check Consistency of Reviewed Article	
	Use hash function to generate a ‘fingerprint’	
	Reviewer sends reviewed article to Chair	
	External Confidentiality [Reviewed Article]	
	Encrypt Reviewed Article	
	Use a 128bits Key (to encrypt article)	
	Chair receives the reviewed articles	
	Internal Confidentiality [Reviewed Article]	
	Authorize Access	
	Chair uses an Identification Key	
	Authenticate Chair with Password	
	Chair uses Single Passoword	
	Uses Access Validation Rules to check if Chair can access this article	
	External Confidentiality [Reviewed Article]	
	Decrypt Article	
	Use a 128bits Key (to decrypt article)	
	Consistency [Reviewed Article]	
	Check Consistency of Article	
	Use DVD midia to make an Article’s copy	
	Chair checks if there is conflict (incompatible evaluations) in reviews of each article	
Constraint:	NFR: Security: *Consistency [Reviewed Article]*, *External Confidentiality [Reviewed Article]*, *Internal Confidentiality [Reviewed Article]* ; User-friendliness	
Exceptions:	If an article has a conflict: (scenario: Conflict Resolution)	

Scenario definition		
Title:	**Conflict Solution**	
Goal:	Conflict be solved.	
Context:	**Open Judgement**	
Geographical Location:	WEB	
Temporal Location:	After articles have been reviewed.	
Constraint:		
Precondition:	Review Article	
Resources:	Computer, Internet, Articles, Reviews	
Constraint:		
Actors:	Chair, Committee Members	
Episodes:	Chair identify an article with a conflict (incompatible evaluations)	
	Chair selects Committee Members in the same area of the article.	
	Chair separates out reviewers of the same institution of the Author.	
	Chair sends reviews with conflict to each selected Committee Member giving the deadline to vote	
	External Confidentiality [Reviewed Article]	
	Encrypt Article	
	Use a 128bits Key (to encrypt article)	
	Committee Members receives the reviews with conflict from Chair	
	External Confidentiality [Reviewed Article]	
	Decrypt Article	
	Use a 128bits Key (to decrypt article)	
	Committee Members votes (to solve the conflict)	
	Committee Members do an **open judgement**	
	Committee Members emit their opinion (vote)	
	Consistency [vote] and External Confidentiality [Vote]	
	Record physically the opinion of the Committee Members in an act	
	Restricts physically the access to the act	
	Chair receives the votes	
Constraint:	NFR: Security, User-friendliness	
Exceptions:		

Scenario definition	
Title:	**Camera Ready Reception**
Goal:	Camera Ready be sent.
Context:	
Geographical Location:	WEB
Temporal Location:	Before the camera ready sending deadline.
Constraint:	
Precondition:	Article be accepted
Resources:	Computer, Internet, camera ready
Constraint:	
Actors:	Author, Chair
Episodes:	*Consistency [Result of Review]*
	Check Consistency of Result of Review
	Use hash function to generate a 'fingerprint' to the Result of Review
	Use DVD midia to make a copy of Result of Review
	Chair sends the result of review to the Author.
	External Confidentiality [Result of Review]
	Encrypt Result of Review
	Use a 128bits Key (to encrypt Result of Review)
	Author receives the result of review
	External Confidentiality [Result of Review]
	Decrypt Result of Review
	Use a 128bits Key (to decrypt Result of Review)
	Author prepares the camera ready.
	Consistency [Camera Ready]
	Authorize Access
	Author uses an Identification Key
	Authenticate Author with Password
	Author uses Single Passoword
	Uses Access Validation Rules to check if Author can make the camera ready
	Check Consistency of Camera Ready
	Use hash function to generate a 'fingerprint' to the Camera Ready
	Author sends the camera ready to Chair.
	External Confidentiality [Camera Ready]
	Encrypt Camera Ready
	Use a 128bits Key (to encrypt Camera Ready)
	Chair receives the camera ready
	External Confidentiality [Camera Ready]
	Decrypt Camera Ready
	Use a 128bits Key (to decrypt Camera Ready)
	Consistency [Camera Ready]
	Check Consistency of Camera Ready
	Use hash function to generate a 'fingerprint' to the Camera Ready
	Use DVD midia to make a copy of Camera Ready
Constraint:	NFR: Security: *External Confidentiality [Result of Review], Consistency [Result of Review] , External Confidentiality [Camera Ready], Consistency [Camera Ready];* User-friendliness
Exceptions:	

Ao final deste passo, os cenários estão enriquecidos com os requisitos não-funcionais refinados e as respectivas medidas de defesa escolhidas para sua

operacionalização. Com o uso de cenários os requisitos gerais são integrados com os requisitos específicos de segurança, em uma única estrutura (os requisitos de segurança são contextualizados em relação aos demais requisitos). A validação dos requisitos não-funcionais, juntamente com a alternativa de operacionalização escolhidas, pode ser feita através da leitura destes cenários detalhados pelos clientes e demais interessados. Com a leitura dos cenários é possível validar:

- o refinamento dos requisitos de segurança;
- a integração entres os requisitos de segurança e os demais requisitos;
- as alternativas de operacionalização escolhidas para cada requisito de segurança.

O uso de cores diferentes para os requisitos não-funcionais (em azul) e para a respectiva alternativa de operacionalização escolhida (em vermelho) auxilia na identificação do desdobramento do requisito não-funcional em si (“o quê”), e sua operacionalização (“como”).

Passo 12 - (Definição de requisitos gerais) Verificação de consistência entre modelos (passo final)

Por fim, é feita a verificação de consistência dos modelos gerados. Este passo visa justamente garantir a qualidade do trabalho como um todo.

4.4 Considerações sobre o Estudo de Caso

O Exercício do estudo de caso foi bastante positivo. Através da aplicação do processo evoluído no *Expert Committee* foi possível perceber melhor como as alterações introduzidas (na evolução do processo) contribuíram para sua melhoria.

O uso de *SDsituations* de fato ajudar a lidar melhor com a complexidade (para análise) dos modelos de i* - *SDsituations* possibilitam abordar o problema da modelagem dos requisitos de segurança através de uma estratégia “dividir para conquistar”.

A introdução de um passo específico para definição de requisitos não-funcionais, especificamente de segurança neste caso, ajudou a organizar melhor a forma de definir estes requisitos: iniciando pela sua elicitação através de catálogos de requisitos, seguida pelos refinamentos por tipos e por tópicos, deixando para a "análise de medidas de defesa" a proposição e escolha das alternativas para operacionalização destes requisitos.

O uso de cenários trouxe como benefícios a integração dos requisitos não-funcionais, especificamente de segurança, e os demais requisitos, apresentando-os em uma forma textual estruturada e com facilitando sua leitura para validação pelos clientes e demais interessados.

Outro ponto percebido através do exercício do estudo de caso foi o uso de heurísticas. Durante a aplicação do processo evoluído, foi preciso definir algumas heurísticas que não estavam muito claras até então, como no caso da identificação de vulnerabilidades. Em outros casos, a definição de mais (ou melhores) heurísticas ainda se faz necessária, como por exemplo, para a identificação de medidas de ataque.

Por fim, concluímos que o exercício de outros estudos de caso deve contribuir para melhoria do processo proposto neste trabalho. A proposição de outros estudos de caso será abordada mais adiante, na seção de trabalhos futuros.

5

Análise de Trabalhos Relacionados

Este capítulo apresenta uma análise de alguns trabalhos relevantes com abordagens distintas para elaboração de requisitos de segurança: “Elaborando requisitos de segurança através da construção de anti-modelos intencionais [10]” e “Casos de Uso de Segurança[11]”. Em seguida é apresentado um trabalho interessante com foco em elicitação de requisitos de segurança: “Elicitação de requisitos baseada em metas com políticas de segurança e privacidade em comércio eletrônico [37]”. Na seção seguinte é apresentado um trabalho mais recente de Liu, Yu e Mylopoulos: “Desenho (Design) de Segurança Baseado em Modelagem Social” [37], seguida de uma breve análise do “Common Criteria [26]”. Finalmente, é apresentado um resumo dos trabalhos analisados juntamente com o nosso trabalho.

5.1 . “Elaborando requisitos de segurança através da construção de anti-modelos intencionais” [10]

5.1.1 Idéia

Este trabalho baseia-se, em termos gerais, na construção, de modo iterativo e concorrente de dois modelos: (i) um modelo intencional do sistema como ele deve ser (*system-to-be*) que cobre tanto o software quanto o ambiente no qual está inserido e inter-relaciona suas metas, agentes, objetos, operações, requisitos e premissas (*assumptions*); (ii) um anti-modelo, derivado do modelo do sistema, que apresenta (através de vulnerabilidades e capacidades requeridas para satisfação de anti-metas) como os elementos do modelo original podem ser maliciosamente ameaçadas, por que e por quem.

Os requisitos de segurança são elaborados sistematicamente pela iteração dos seguintes passos: (a) instanciar de padrões associados a classes (sub-tipos) de requisitos de segurança como privacidade, integridade, autenticação; (b) derivar especificações do anti-modelo que ameacem as especificações do modelo; e (c) derivar contra-medidas alternativas para estas ameaças e definir novos requisitos através da seleção das alternativas que melhor atendam a outros requisitos de qualidade do modelo. O modelo original é então enriquecido com novos requisitos de segurança derivados como contra-medidas para o anti-modelo.

5.1.2 Background

O método é baseado em um *framework* orientado a metas [19], desenvolvido anteriormente pelo próprio Lamsweerde, para geração e resolução de obstáculos ao atendimento de requisitos e foi estendido para obstáculos maliciosos, permitindo que o processo de refinamento de obstáculos seja guiado pelas metas dos atacantes com o objetivo de derivar vulnerabilidades observáveis e ameaças que possam vir a serem implementadas. Os conceitos de *metas*, *agentes* e *papéis* existentes neste *framework* são similares aos de i*.

Neste método, metas são organizadas em hierarquias E/OU de refinamento de abstração onde as metas de nível mais alto são em geral estratégicas e envolvem vários agentes e as metas de nível mais baixo são em geral técnicas e envolvem poucos agentes. O refinamento de uma meta termina quando cada uma de suas sub-metas é passível de realização por algum agente individual. Uma meta é operacionalizada em especificações de operações necessárias para alcançá-la. *Metas flexíveis* prescrevem comportamentos preferenciais, e podem ser usadas para selecionar alternativas preferenciais em um grafo de refinamento de meta E/OU.

Propriedades do domínio são declarações descritivas sobre o ambiente, como leis físicas ou normas organizacionais.

Um *requisito* é uma meta terminal sob responsabilidade de algum agente no modelo original. Uma *expectativa* é uma meta terminal de um agente no ambiente (diferentemente dos requisitos, expectativas não podem ser realizadas pelo sistema). Um *atacante* é um agente malicioso no ambiente. Uma *anti-meta* é uma

meta de um atacante. Uma *ameaça* é um obstáculo intencionalmente estabelecido por um atacante. *Objetos* têm estados definidos por seus atributos e relacionamentos para outros objetos. Podem ser passivos (entidades, associações, eventos) ou ativos (agentes).

Um *obstáculo* a uma meta, em termos declarativos, é uma condição cuja satisfação pode evitar que a meta seja alcançada (satisfeita). A análise de obstáculos consiste em adotar uma visão pessimista para as metas, requisitos e expectativas que são elaboradas. O princípio é identificar tantos modos de obstruir metas, requisitos e expectativas quanto possíveis. Para resolver tais obstruções são elaborados requisitos mais completos para sistemas mais robustos.

A especificação de requisitos de segurança específicos da aplicação é obtida, através da aplicação de padrões de especificação genéricos associados às classes de segurança, da seguinte forma: (a) instanciando meta-classes como Objeto, Agentes e atributos genéricos como informação para classes sensíveis específicas da aplicação, e (b) especializando predicados como ‘Autorizado’, ‘Sob Controle’ ou ‘Usando’ pela substituição por definições específicas da aplicação.

Uma árvore de obstáculos é ancorada por meta e sua raiz é a negação de uma meta (do modelo original). Os obstáculos podem ser gerados formalmente pela regressão da negação através das propriedades do domínio e outras declarações de meta ou pelo uso de obstrução formal e refinamento de padrões.

5.1.3 Procedimento

Os passos do procedimento para construção do anti-modelo são os seguintes:

1 – Capturar as anti-metas iniciais através da negação dos padrões de especificação de metas de classe de segurança (confidencialidade, integridade, disponibilidade), instanciados para objetos sensíveis do modelo de objetos;

2 – Para cada anti-meta, elicitar potenciais atacantes que poderiam possuir a anti-meta;

3 – Para cada anti-meta e (para cada) classe de atacante correspondente identificada, elicitar as anti-metas de mais alto nível de abstração;

4 – Elaborar o grafo E/OU da anti-meta através de refinamento “E” (*AND*) de anti-metas por todos os ramos alternativos, com o objetivo de derivar anti-metas terminais passíveis de realização pelo atacante identificado ou por um agente do sistema atacado. Os primeiros são anti-requisitos atribuídos aos atacantes enquanto os últimos são vulnerabilidades atribuídas aos agentes do sistema.

5 – Derivar os anti-modelos de agentes e objetos da especificação de anti-metas.

6 – Operacionalizar (via refinamento E/OU) todos os anti-requisitos em termos das capacidades do correspondente atacante.

Uma vez gerados os anti-modelos intencionais, o próximo passo é considerar contra-medidas alternativas para as vulnerabilidades e anti-requisitos encontrados. Estas contra-medidas são geradas sistematicamente usando operadores como: substituição de meta; substituição de agente; enfraquecimento de meta; restauração de meta; e prevenção de anti-meta; juntamente com operadores específicos de segurança como: proteger de vulnerabilidade; neutralizar de ameaça; e evitar vulnerabilidade.

5.1.4 Semelhanças e diferenças de [10] com o nosso trabalho e [3]

A construção de anti-modelos proposta por Lamsweerde [10] tem semelhança com os passos 1 – Identificação de atacantes, 3 – Identificação de intenções maliciosas e 4 – identificação de medidas de ataque da análise de medidas de ataque; de nossa abordagem. (estes passos foram inicialmente propostos em [3]).

Anti-metas correspondem às intenções maliciosas, embora em [10] as anti-metas iniciais sejam formalmente geradas a partir da negação das metas flexíveis de segurança. Uma diferença em relação à ordem de execução dos passos é que em [10], são identificados possíveis atacantes para uma anti-meta, enquanto na

nossa abordagem são identificadas possíveis intenções maliciosas para um atacante.

A especificação dos requisitos em [10] é feita através da aplicação de padrões de especificação, que guarda semelhança com catálogos de requisitos não-funcionais, embora [10] refine esses padrões formalmente. A operacionalização de anti-metas (medidas de ataque) em [10] também podem ser feita formalmente.

A escolha entre as alternativas é guiada pela prioridade dos demais requisitos de qualidade (embora não fique claro em [10] como estes outros requisitos são afetados por, ou afetam, estas escolhas).

Uma diferença relevante é que requisitos funcionais e não-funcionais são modelados da mesma forma em [10] enquanto em i* os requisitos não-funcionais (metas flexíveis) distinguem-se claramente dos demais requisitos.

5.2 “Casos de Uso de Segurança” [11]

5.2.1 Idéia

A idéia geral é que casos de uso de segurança (*security use cases*), usados para analisar e especificar requisitos de segurança, podem ser derivados dos casos de mau uso (*misuse cases*), estes últimos usados para analisar e especificar ameaças. Assim como casos de uso (*use cases*) e casos de mau uso, os casos de uso de segurança são apresentados de forma descritiva, numa seqüência de passos. O rastro dos requisitos pode ser feito nos dois sentidos entre casos de uso, casos de mau uso e casos de uso de segurança.

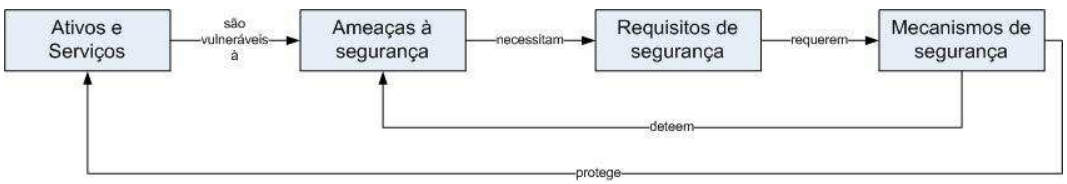

Figura 5.1 - Abordagem de casos de uso de segurança, linhas gerais, capturada de [11]

5.2.2 Semelhanças e diferenças de [11] com o nosso trabalho e [3]

No tipo de abordagem usado em [11], baseadas em anti-requisitos, a questão intencional não vem à tona. Tanto as motivações dos atacantes, quanto as motivações dos atores do sistema não são abordadas. Estas questões são centrais tanto no nosso trabalho e em [3], como também em [10].

Não está claro em [11] como os requisitos de segurança são desdobrados em mecanismos de segurança, nem há menção a escolha de alternativas para 'satisfação a contento' dos requisitos de segurança, deixando abertas lacunas mencionadas por Giogini em [28]: a importância de se capturar requisitos de segurança de alto nível, e de tornar claros os motivos do uso de mecanismos de proteção como criptografia e autorização de acesso.

Estas lacunas prejudicam o rastro dos requisitos de segurança e deixam encoberto o *rationale* da escolha de alternativas.

A apresentação em forma descritiva dos requisitos de segurança, juntamente com os demais requisitos, aparece em [11] em virtude dos casos de uso de segurança serem descritos de forma parecida a de casos de uso e casos de mau uso. Em nosso trabalho, esta forma descritiva aparece em virtude da introdução do passo 5 – Definição de cenários.

5.3 "Elicitação de requisitos baseada em metas com políticas de segurança e privacidade em comércio eletrônico" [37]

5.3.1 Idéia

Este trabalho segue uma filosofia semelhante aos demais trabalhos com abordagem orientada a metas. O processo usado, em linhas gerais, é composto dos seguintes passos:

1. identificação dos atores envolvidos (*stakeholders*);
2. elicitação das metas do sistema;

3. atribuição de valores para as metas;
4. refinamento das metas;
5. operacionalização das metas;
6. avaliação de riscos;
7. criação de políticas de segurança e privacidade;
8. avaliação de conformidade com políticas existentes.

5.3.2 Semelhanças e diferenças

Este trabalho traz como principais diferenciais em relação aos demais trabalhos com a mesma abordagem a atribuição de valores para as metas e a avaliação de riscos, e em menor grau a avaliação de conformidade com políticas existentes.

A avaliação de riscos é dividida em três sub-atividades: (i) identificação de ameaças e vulnerabilidades; (ii) estimativa da probabilidade de ocorrência e (iii) escolha da estratégia de mitigação do risco. A sub-atividade de identificação de ameaças e vulnerabilidade responde também pela identificação das fontes de ameaça (atacantes), motivação (intenções maliciosas) e ações (medidas de ataque). A estimativa da probabilidade de ocorrência é feita de forma qualitativa, usando uma escala com três valores: alto, médio e baixo. A escolha da estratégia de mitigação do risco é feita pela adição de uma nova meta ou sub-meta em resposta ao risco identificado ou através de um novo refinamento de meta visando adicionar uma restrição ou atenuar o risco identificado. Embora não esteja muito claro, essa meta serve para definição dos requisitos não-funcionais (modelados como metas), em função dos riscos identificados.

A partir deste ponto (após a avaliação de riscos) há dois caminhos a serem seguidos: avaliação da conformidade com as políticas de segurança e privacidade da organização, caso elas existam, com o objetivo de garantir que os requisitos sejam implementados de acordo com estas políticas; ou, a criação de novas políticas de segurança e privacidade para implementação destes requisitos.

As políticas de segurança e privacidade neste trabalho têm significados semelhantes ao que as alternativas de operacionalização têm no nosso: políticas de segurança e privacidade são normas de como os requisitos de segurança devem ser implementados, como por exemplo, políticas de identificação e autenticação.

Este trabalho traz alguns pontos interessantes não explorados em nosso trabalho. A atribuição de valores às metas é bastante interessante, e pode ser usada na priorização dos requisitos não-funcionais para escolha da alternativa de operacionalização. A avaliação de riscos, com a atribuição de um valor para o risco (ou para ameaça) é outra idéia interessante. O valor para o risco fornece um parâmetro para avaliação, ainda que de forma subjetiva ou qualitativa, do quanto se deseja investir para garantir a segurança. Em alguns casos, em virtude do impacto proporcionado e da probabilidade do risco vir a se concretizar, pode ser considerado satisfatório apenas "impedir parcialmente" (*weakly deny*) a satisfação da intenção maliciosa de algum atacante em vez de "impedir completamente" (*deny*) a sua satisfação.

5.4 "Desenho (*Design*) de Segurança Baseado em Modelagem Social" [37]

Neste trabalho mais recente (publicado em 2006) Liu, Yu e Mylopoulos apresentam, de forma mais resumida, um processo idêntico ao apresentado em [3]. Este processo, focado na definição dos requisitos específicos de segurança, é composto pelos seguintes passos: (i) identificação de atores e objetivos; (ii) identificação de atacantes; (iii) avaliação das conseqüências de ataques; e (iv) identificação de contramedidas alternativas.

O passo (ii) identificação de atacantes é mais elaborado que em [3], com o estabelecimento de um par de fronteiras de confiança para o sistema: uma interna e outra externa. A idéia é que os atores dentro da fronteira interna de confiança, ou além da fronteira externa, não constituem fontes de ameaça ao sistema uma vez que atores dentro dos limites destas fronteiras são considerados confiáveis. Com isto o 'espaço de procura' por atacantes é delimitado por estas fronteiras. Em

nossa proposta não usamos fronteiras de confiança para delimitar o ‘espaço de busca’ por possíveis atacantes.

Nossa abordagem apenas segmenta os atacantes em “internos” e “externos”. São considerados atacantes “internos” os atacantes derivados dos atores legítimos, com as mesmas habilidades e capacidades, porém com intenções maliciosas; enquanto os demais atacantes, que possam ameaçar o sistema sem possuir capacidades e habilidades semelhantes aos atores legítimos são considerados atacantes “externos”.

Tanto neste novo trabalho, como em [3], Liu, Yu e Mylopoulos consideram a possibilidade de ataques acidentais como, por exemplo, mau funcionamento de software, desentendimento de instrução dos usuários ou execução imprópria de instruções. Os ataques acidentais seriam ataques não intencionais.

No nosso trabalho, preferimos abordar o assunto de forma um pouco diferente, não considerando este tipo de acidente como ataque, justamente por não ser intencional. Porém, do mesmo modo que Liu, Yu e Mylopoulos, nos preocupamos em observar as vulnerabilidades do sistema a estes tipos de acidentes.

Os demais passos apresentados em [37] são semelhantes aos apresentados em [3], discutidos anteriormente no capítulo 3.

5.5 “Common Criteria” [26]

O “Common Criteria” não é um trabalho na linha de definição de requisitos de segurança, uma vez que não propõe como elicitar, modelar e analisar estes requisitos. É um trabalho voltado mais especificamente para uma sub-atividade da análise: a verificação. O “Common Criteria” propõe uma série de critérios para avaliação da segurança de um software. Estes critérios, muitas vezes, são alternativas para operacionalização de requisitos de segurança que um software ou

componente deve ter para ser considerado seguro, como por exemplo, o uso de mecanismos de autenticação e criptografia.

Em nossa avaliação, este tipo de trabalho é bastante útil para atividade de verificação, ou ainda como fonte para elicitação de alternativas de operacionalização para requisitos de segurança de software. No entanto, deixa em aberto lacunas da modelagem (como os requisitos não-funcionais são modelados, como se integram aos demais requisitos) e não é muito claro em relação à escolha das alternativas disponíveis.

5.6 Resumo

A tabela 5.1, a seguir, apresenta um resumo das principais características dos trabalhos analisados neste capítulo e de nosso trabalho. Estes trabalhos são frutos de um esforço relativamente recente da comunidade acadêmica sobre em tratar os requisitos de segurança durante a atividade de modelagem.

Resumo de características dos trabalhos analisados			
Trabalho Analisado	**Ponto de partida**	**Idéia chave**	**Diferencial**
Elaborando requisitos de segurança através da construção de anti-modelos intencionais [10]	Anti-metas geradas através da negação das metas originais	Construção de anti-modelos para os anti-requisitos a partir da negação do modelo de requisitos	A geração de obstáculos para a construção do anti-modelo pode ser feita formalmente.
Casos de Uso de Segurança [11]	"Casos de mau uso" (*misuse cases)*	Os casos de uso de segurança são derivados a partir dos "casos de mau uso"	Expressos de forma descritiva, semelhante aos casos de uso
Elicitação de requisitos baseada em metas com políticas de segurança e privacidade em comércio eletrônico [37]	Elicitação das metas dos atores	As metas são elicitadas e refinadas até sua operacionalização, avaliando-se a conformidade da operacionalização com as políticas de segurança existente ou criando-se novas políticas	Atribuição de valor às metas, tratamento de riscos mais apurado e uso de políticas de segurança.
Desenho (*Design*) de Segurança Baseado em Modelagem Social [37]	Elicitação das metas dos atacantes (intenções maliciosas)	Construção de modelos intencionais, incluindo atacantes e suas metas	Estabelecimento de um par de fronteiras de confiança para restringir o espaço de busca por atacantes
Common Criteria [26]	Critérios estabelecidos na própria norma	Conjunto de boas práticas e critérios para aferição de segurança	Foco na atividade de verificação.
Uso de estratégias orientadas a metas para modelagem de requisitos de segurança	Elicitação das metas dos atacantes (intenções maliciosas)	Construção de modelos intencionais, incluindo atacantes e suas metas	Uso de *SDsituations* para lidar com a complexidade dos modelos e cenários para apresentação dos requisitos de segurança.

Tabela 5-1 – Resumo de características dos trabalhos analisados

6
Conclusão

Este capítulo apresenta uma avaliação da aplicação da abordagem proposta no nosso trabalho, as nossas contribuições e propostas para trabalhos futuros.

6.1 Avaliação

A aplicação do processo evoluído (Figura 3.3) proposto em nosso trabalho mostrou-se satisfatória no estudo de caso do *Expert Committe*, alcançando os objetivos traçados inicialmente, principalmente documentar o *rationale* da escolha da alternativa para operacionalização e possibilitar o rastro dos requisitos de segurança de alto nível até sua operacionalização (como são refinados em requisitos de nível de abstração menor; como são operacionalizados).

O objetivo de documentar o *rationale* da escolha da alternativa para operacionalização dos requisitos de segurança foi alcançado através dos modelos elaborados durante a execução do Passo 5 (Análise de Segurança) – Análise de medidas de defesa, apresentada na seção 4.2.

O rastro desde os requisitos de segurança de alto nível até sua operacionalização é possível, como pode ser verificado no exemplo do rastro do requisito de segurança do artigo revisado apresentado na tabela 6.1, a seguir.

Os resultados apresentados pela aplicação no estudo de caso se mostraram positivos. Acreditamos que também seja possível obter resultados positivos na aplicação do processo em outros casos.

Rastro do requisito: Segurança do Artigo Revisado	
Refinamento:	- **Confidencialidade interna** [Artigo Revisado]; - **Confidencialidade externa** [Artigo Revisado]; - **Consistência** [Artigo Revisado]
Vulnerabilidades relacionadas:	- Dependência de recurso Artigo Revisado (Organizador da Conferência x Autor); - Dependência de tarefa Revisar Artigo (Organizador da Conferência x Revisor); - Dependência de recurso Revisões com Conflito (Resolvedor de Conflito x Organizador da Conferência); - Dependência de recurso Resultado da Revisão (Autor x Organizador da Conferência).
Aplicação (Situações/Cenários):	Revisão de Artigos (‘*Article Reviews*’); Resolução de Conflitos (‘*Conflicts Solution*’); Recepção das Versões Finais (‘*Camera Ready Reception*’)
Atacantes:	- Atacante do Revisor; - Atacante do *Chair*; - Atacante do Autor; - Atacante do Membro do Comitê;
Intenções maliciosas dos atacantes:	- Revisões serem fraudadas (Atacante do Revisor, Atacante do *Chair*); - Revisões serem roubadas (Atacante do Revisor, Atacante do *Chair*, Atacante do Membro do Comitê) - Informação ser corrompida (Vírus, Internet Cracker); - Informação ser descoberta (Internet Hacker)
Alternativas de operacionalização p/ **Confidencialidade Interna**:	Usar chave de identificação; Usar regra de validação de acesso; Autenticar com senha - Autenticar com senha simples; - Autenticar com múltiplas senhas;
Alternativa escolhida p/ **Confidencialidade Interna**:	Usar chave de identificação; Usar regra de validação de acesso; Autenticar com senha simples;
Justificativa da escolha p/ **Confidencialidade Interna**:	Não ferir a usabilidade
Alternativas de operacionalização p/ **Confidencialidade Externa**:	Criptografar - Usar chave de 64Bits - Usar chave de 128Bits
Alternativa escolhida p/ **Confidencialidade Externa**:	Usar chave de 128Bits
Justificativa da escolha p/ **Confidencialidade Externa**:	Priorizar Segurança em relação ao Desempenho
Alternativas de operacionalização p/ **Consistência**	Usar chave de identificação; Usar regra de validação de acesso; Autenticar com senha - Autenticar com senha simples; - Autenticar com múltiplas senhas; Usar função *hash*; Informação ser copiada: - Usar mídia de DVD (para cópia); - Usar mídia magnética;
Alternativa escolhida - **Consistência**:	Usar chave de identificação; Usar regra de validação de acesso; Autenticar com senha simples; Usar função *hash*; Usar mídia de DVD (para cópia)
Justificativa da escolha p/ **Consistência**:	Não ferir a usabilidade

Tabela 6-1 - Rastro do requisito: Segurança do Artigo Revisado

Embora o estudo de caso tenha sido focado apenas nos requisitos de confidencialidade e consistência, acreditamos que a abordagem proposta possa ser aplicada também para requisitos de disponibilidade, com alguns cuidados especiais que discutiremos na seção trabalhos futuros, mais adiante.

6.2 Contribuições

A principal contribuição de nosso trabalho foi a evolução do processo proposto em [3], com a inclusão de passos específicos definição de SDsituation, definição de cenários, refinamento de atores (legítimos e atacantes) e definição de requisitos não-funcionais. Além dos passos em si, nosso trabalho propôs o uso de heurísticas para elaboração destes novos passos, como as heurísticas propostas em [4] para definição de SDsituations e definição de cenários, as propostas em [7] para o refinamento de requisitos não-funcionais, e as heurísticas propostas em nosso trabalho para o refinamento de atores.

Com estas contribuições, no estudo de caso conseguimos lidar melhor com a complexidade (para análise) dos modelos produzidos pelo framework i*, usando uma estratégia “dividir para conquistar”; apresentamos um entendimento melhor dos atores e suas relações; tornamos explícito o processo de definição dos requisitos não-funcionais, em especial, os de segurança; e, apresentamos os requisitos através de cenários, que evoluíram ao longo do processo desde a versão inicial (elaborada na primeira execução do passo 5, onde os requisitos de segurança apareciam apenas como restrições destes cenários) até a versão detalhada apresentada ao final do processo (onde os cenários apresentavam os requisitos de segurança integrados aos demais requisitos com as respectivas escolhas de operacionalização). Acreditamos que estes cenários detalhados representem uma forma mais adequada para validação pelos usuários e demais interessados que os modelos SD e SR.

Por fim, acreditamos que estes benefícios possam ser obtidos também na aplicação do processo evoluído em casos reais e em outros estudos de caso.

6.3 Trabalhos Futuros

Como trabalhos futuros, esperamos aplicar o processo evoluído, apresentado no capítulo 3, para elaboração dos requisitos de segurança em casos reais e outros estudos de caso com o objetivo de validá-lo.

Acreditamos que seja possível aplicar o processo evoluído também à elaboração de requisitos de disponibilidade. Para tanto, é necessário modelar cuidadosamente atores e recursos de infra-estrutura presentes no ambiente, uma vez que a disponibilidade de uma aplicação dependente destes atores e recursos. Por exemplo, a aplicação depende de recursos de hardware para ser rodada; de recursos a serem disponibilizados e/ou tarefas a serem desempenhadas pelo sistema operacional, por servidores de aplicação e por servidores *web*; e, de uma infra-estrutura de rede que possibilite a comunicação entre os atores. Consequentemente, a disponibilidade dos recursos, das tarefas, satisfação de metas e 'satisfação a contento' de metas flexíveis (acordadas com outros atores) dependerá sensivelmente da infra-estrutura do ambiente em que a aplicação se encontre.

O desenvolvimento futuro de uma ferramenta de modelagem integrada é necessário para viabilizar o uso eficiente do processo proposto. Em geral, os elementos usados como nós nos modelos i* (atores, metas, tarefas e recursos) aparecem em mais de um diagrama. Uma ferramenta integrada pode contribuir imensamente para manutenção da consistência dos modelos gerados.

Outro trabalho futuro a ser desenvolvido é modelar os requisitos de segurança com o objetivo de transformá-los em aspectos, justamente por serem os requisitos de segurança (e sua operacionalização) candidatos naturais a aspectos.

7
Referências

[1] YU, E. **Modelling Strategic Relationships for Process Reengineering**, PhD Thesis, Graduate Department of Computer Science, University of Toronto, Toronto, Canada, 1995, Pag. 124.

[2] SIMON, H. **Theories of Decision-Making in Economics and Behavioral Science**. The American Economic Review, Vol. 49, No. 3, Pags. 253-283

[3] LIU, L.; YU. E.; MYLOPOULOS, J. **Security and Privacy Requirements Analysis within a Social Setting**. Proceedings of the International Conference on Requirements Engineering (RE'03). Monterey, California, September 2003. Pags. 151-161.

[4] OLIVEIRA, A.; CYSNEIROS, L. **Defining Strategic Dependency Situations in Requirements Elicitation**. Proceedings of IX Workshop on Requirements Engineering (WER'2006). Rio de Janeiro, Brasil, Julho 2006. Pags. 12-23

[5] LEITE, J.C.S.P. et al. **A Scenario Construction Process.** RE Journal: Vol 5 N. 1, Springer-Verlag London Limited. 2000. Pags. 38-61

[6] LEITE, J.C.S.P.; FRANCO, A.; Client A **Strategy for Conceptual Model Acquisition**. Proceedings of the Internaional Symposium on Requirements Engineering, IEEE Computer society Press, San Diego. 1993. Pags 243-246.

[7] CHUNG, L. et al. **Non-Functional Requirements in Software Engineering**. Kluwer Academic Publishers. Boston/Dordrecht/London. 2000

[8] CYSNEIROS,L.M.; LEITE, J.C.S.P; NETO, J.S.M. **A Framework for Integrating Non-Functional Requirements into Conceptual Models**. RE Journal: Vol 6 , N. 2, Springer-Verlag London Limited. 2001. Pags. 97-115.

[9] MANADHATA, P.; WING, J. M. **Measuring a System's Attack Surface**. Technical Report CMUCS-04-102, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA. 2004

[10] VAN LAMSWEERDE, A. **Elaborating security requirements by construction of intentional anti-models**. Proceedings. 26th International Conference on Software Engineering, 2004. ICSE 2004. Pags. 148- 157.

[11] FIRESMITH, D. **Security Use Cases**. Journal of Object Technology, vol. 2, no.3, May-June 2003, pp. 53-64.

[12] SINDRE, G.; OPDAHL, A.L. **Eliciting security requirements by misuse cases.** Proceedings. 37th International Conference on Technology of Object-Oriented Languages and Systems, 2000. TOOLS-Pacific 2000. Pags. 120-131.

[13] LEITE, J.C.S.P. et al. **Understanding the Strategic Actor Diagram: An Exercise of Meta Modeling**. WER 2007 – 10th Workshop on Requirements Engineering, Toronto, Canada, 2007 – **aceito.**

[14] AVIZIENIS, A. et al. **Basic concepts and taxonomy of dependable and secure computing**. IEEE Transactions on Dependable and Secure Computing, 2004. Pags 11-33

[15] WEINSTOCK, C.; GOODENOUGH, J.; HUDAK, J. **Dependability Cases**. Technical Note CMU/SEI-2004-TN-016, Proceedings of the International Conference on Dependable Systems and Networks. 2004.

[16] LIU, L.; YU, E. **Intentional Modeling to Support Identity Management**. Proceedings of the 23rd International Conference on Conceptual Modeling (ER 2004). 2004. Pags. 555– 566

[17] OLIVEIRA, A. et al. **Integrating Scenarios, i*, and AspectT in the Context of Multi-Agent Systems**. Cascon 2006 the 16th Annual International Conference on Computer Science and Software Engineering.

[18] ANTÓN, A. **Goal Based Requirements Analisys.** Second International Conference on Requirements Engineering (ICRE'96). 1996. Pags. 136-144

[19] VAN LAMSWEERDE, A.; LETIER, E. **Handling Obstacles in Goal Oriented Requirements Engineering.** IEEE Transaction on Software Engineering, Specila Issue on Exception Handling, Vol. 26 No. 10. 2000. Pags 978-1005

[20] CYSNEIROS, L.; YU, E.; LEITE, J.C.S.P. **Cataloguing Non Funcional Requirements as Softgoals Networks**. Proceedings of Requirements Engineering for Adaptable Architectures - 11th International Requirements Engineering Conference. 2003. Pags.13-20

[21] DEVANBU, P.; STUBBLEBINE, S. **Software Engineering for Security: a Roadmap.** The Future of Software Engineering. Special volume of the proceedings of the 22nd International Conference on Software Engineering -- ICSE 2000, June 2000. 23

[22] ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. **NBR ISO/IEC 17799: Tecnologia da informação – código de prática para a gestão da segurança da informação.** Rio de Janeiro. 2001.

[23] CERT. Programa do Instituto de Engenharia de Software (SEI) - Carnegie Mellon University. Apresenta uma série de estatísticas relacionadas com trabalhos desenvolvidos na área de segurança de software. Disponível em: <http://www.cert.org/stats/cert_stats.html>. Acessado em: 03 fev. 2007.

[24] ALEXANDER, L. **Misuse Cases: Use Cases with Hostile Intent**. IEEE Software, Vol. 20. 2003. Pags. 58-66

[25] VAN LAMSWEERDE, A. et al. **From System Goals to Intruder Anti-Goals: Attack Generation and Resolution for Security Requirements Engineering.** Proceedings of the 2nd International Workshop on Requirements for High Assurance Systems, 2003. Pags. 49-56

[26] INTERNATIONAL ORGANIZATION FOR STANDARDIZATION/ INTERNATIONAL ELECTROTECHNICAL COMMISSION. **ISO/IEC 15408-2:2005(E): Information technology - Security techniques - Evaluation criteria for IT security - Part 2: Security functional requirements**. Genebra, 2005.

[27] CROOK, R. et al. **Security Requirements Engineering: When Anti-Requirements Hit the Fan.** Proceedings of IEEE International Requirements Engineering Conference (RE'02). Essen, Germany. 2002. Pags. 9-13

[28] GIOGINI, P.; MASSACCI, F.; ZANNONE, N. **Security and Trust Requirements Engineering**. FOSAD 2005 - Foundations of Security Analysis and Design III. 2005. Pags. 237-272

[29] THAYER, R; DORFMAN, M. **System and Software Requirements Engineering**. IEEE Computer Society Press, 1990.

[30] LEITE, J.C.S.P. **Engenharia de Requisitos - Notas de Aula**. Disponível em: <http://engenhariaderequisitos.blogspot.com/2006/08/processo.html> Acessado em: 14/03/2007 às 16h38min.

[31] LEITE, J.C.S.P. **Engenharia de Requisitos - Notas de Aula**. Disponível em: <http://engenhariaderequisitos.blogspot.com/2006/08/elicitao.html>. Acessado em: 14/03/2007 às 16h14min.

[33] LEITE, J.C.S.P. **Engenharia de Requisitos - Notas de Aula**. Disponível em: <http://engenhariaderequisitos.blogspot.com/2006/10/aula-19.html>. Acessado em: 14/03/2007 às 16h14min.

[34] DESCRIÇÃO do *Expert Committee*. Disciplina Engenharia de Software de Sistemas Multi-Agentes, Curso de mestrado em Informática, Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro. 1º Semestre de 2005. Disponível em: http://web.teccomm.les.inf.puc-rio.br/uploads/7/74/ExpertCommittee3.doc

[35] GONZALEZ-BAIXAULI, B.; LEITE, J.C.S.P.; MYLOPOULOS, J. **Visual Variability Analysis for Goal Models**. 12th IEEE International Requirements Engineering Conference (RE'04), 2004. Pags 198-207

[36] ROCHA, S.; ABDELOUAHAB, Z.; FREIRE, E. **Requirement Elicitation Based on Goals with Security and Privacy Policies in Electronic Commerce.** Anais do WER05 - Workshop em Engenharia de Requisitos, Porto, Portugal. 2005. Pags 63-74

[37] LIU, L.; YU, E.; MYLOPOULOS, J. **Security Design Based on Social Modelling**. 30th Annual International Computer Software and Applications Conference (COMPSAC'06). 2006. Pags. 71-78

[38] PROJECT MANAGEMENT INSTITUTE. **ANSI/PMI 99-001-2004**: **A Guide to the Project Management Body of Knowledge, 3th Edition.** Project Management Institute, Inc. 2004

Livros Grátis

(http://www.livrosgratis.com.br)

Milhares de Livros para Download:

Baixar livros de Literatura
Baixar livros de Literatura de Cordel
Baixar livros de Literatura Infantil
Baixar livros de Matemática
Baixar livros de Medicina
Baixar livros de Medicina Veterinária
Baixar livros de Meio Ambiente
Baixar livros de Meteorologia
Baixar Monografias e TCC
Baixar livros Multidisciplinar
Baixar livros de Música
Baixar livros de Psicologia
Baixar livros de Química
Baixar livros de Saúde Coletiva
Baixar livros de Serviço Social
Baixar livros de Sociologia
Baixar livros de Teologia
Baixar livros de Trabalho
Baixar livros de Turismo