3
Estratégia de Validação Baseada em Simulação

Após apresentarmos o *framework* i* na seção anterior, pôde–se observar a complexidade que essa modelagem carrega, e o quão seria difícil validar esses diagramas diretamente com os interessados.[1]

Fundamentado neste ponto, este capítulo tem por objetivo detalhar uma estratégia baseada em simulação para validação de modelos em i*. Esta estratégia tem por finalidade encapsular a complexidade inerente dos modelos i* para os interessados, e ao mesmo tempo colher discrepâncias, erros ou omissões que possam estar presentes nos modelos construídos pelo engenheiro de requisitos.

Para tal, utilizaremos um exemplo abordado por Oliveira em [2], de um sistema de controle de caixa de restaurante, para tornar mais simples e didática a explicação da estratégia.

3.1. Visão Geral da Estratégia

A estratégia proposta tem por objetivo auxiliar o engenheiro de requisitos a validar os modelos i* junto aos interessados (*stakeholders*), utilizando para isso técnicas interativas para minimizar a complexidade intrínseca dos modelos a serem validados. Além disso, ajustar o modelo proposto do domínio do problema e corrigir falhas apontadas pelos interessados durante o processo de validação. As técnicas interativas supracitadas envolvem simulação [8] [9] [10] e registro de impressões em áudio e vídeo durante a realização da simulação [11].

O modelo SADT[2] [12] a seguir descreve a visão geral da estratégia.

[1] Este trabalho trata s*takeholders* como interessados.

[2] Notação do SADT: caixas representam atividades, setas à esquerda representam as entradas das atividades, setas à direita as saídas, setas acima representam mecanismos e setas abaixo representam controles.

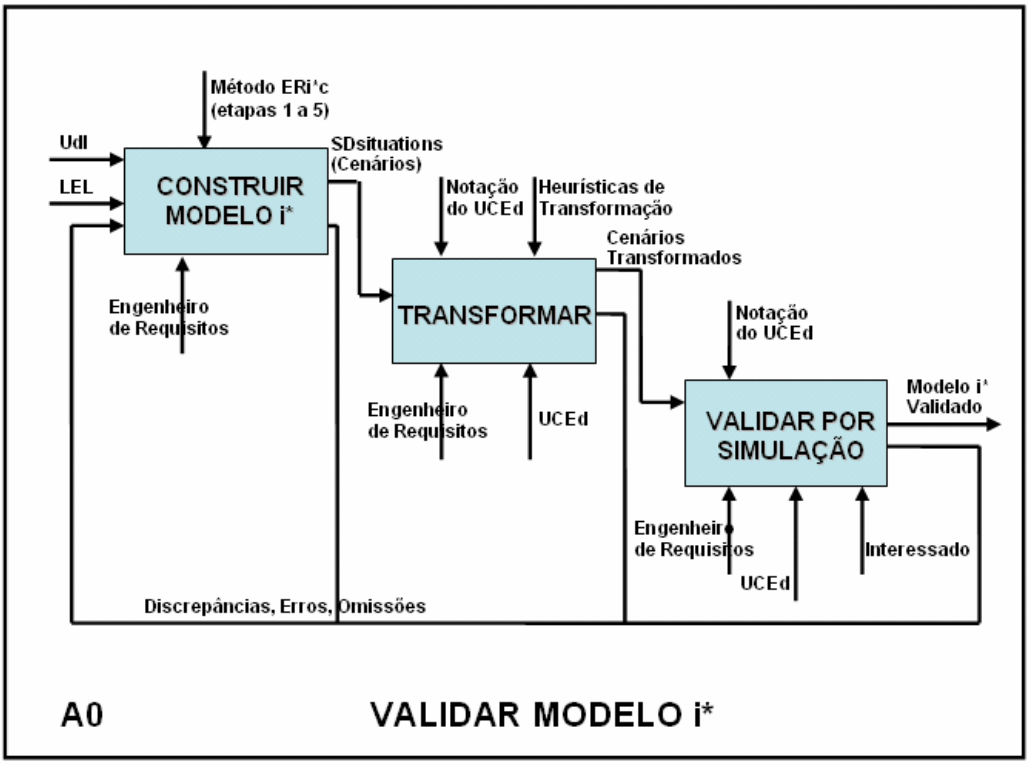

Figura 3.1 – SADT da estratégia de validação de modelos em i*.

O modelo mencionado apresenta as seguintes atividades:

1. CONSTRUIR MODELO i*
2. TRANSFORMAR
3. VALIDAR POR SIMULAÇÃO

A estratégia se inicia com a construção dos modelos em i*. Essa construção de modelos segue as etapas 1 a 5 método ERi*c – Engenharia de Requisitos Intencional, proposto por Oliveira em [2]. A escolha do método ERi*c se baseia no fato da utilização do conceito de intencionalidade, assim como no *framework* i*. Além disso, o método ERi*c oferece uma estratégia *bottom–up* simples, fundamentada no léxico estendido da linguagem (LEL). A estratégia de simulação permite a validação dos modelos i* na etapa 6 (seis) do método ERi*c.

Uma vez construídos os modelos i*, torna–se necessário transformá–los para que estes possam ser validados. Neste ponto, sem alguma transformação, seria uma tarefa quase impossível validar os diagramas em i* junto aos

interessados, devido à complexidade intrínseca desses modelos. Sendo assim, o engenheiro de requisitos deve ter em mãos as descrições em cenários refletindo as *SDsituations* elicitadas, e transformá–las, seguindo um processo bem definido, para servir como entrada da ferramenta de simulação. Este trabalho usa especificamente a ferramenta *UCEd* [8] [9] [16].

O *Use Case Editor* (*UCEd*) é uma ferramenta que permite a construção e edição de casos de uso, a geração de seus respectivos diagramas de estados e a simulação dos mesmos, através de uma interface gráfica amigável, provendo um suporte automatizado para o engenheiro de requisitos.

O *UCEd* foi escolhido para este trabalho por ser um software livre e por possuir as características supracitadas, importantes e únicas comparado a maioria das ferramentas de modelagem de requisitos existentes.

Por fim, as descrições dos modelos em cenários serão simuladas na ferramenta para que o interessado possa validá–las junto ao engenheiro de requisitos. Nesta parte será colhido o *feedback* fornecido pelo interessado e serão ajustadas possíveis discrepâncias, erros e/ou omissões nos modelos i* construídos.

3.2. Construir Modelos i*

A construção dos modelos i* será realizada utilizando as etapas de 1 a 5 do método ERi*c – Engenharia de Requisitos Intencional. Este método fornece algumas contribuições importantes durante o processo de construção, como as situações de dependência estratégica (*SDsituations*), a técnica AGFL (*Agent Goals From Lexicon*) para elicitar metas dos atores, painéis de intencionalidade (diagramas IP), entre outras contribuições.

Além disso, é um método *bottom–up* simples de empregar, possui como âmago o conceito de intencionalidade, que reflete as motivações e interesse dos atores, assim como o *framework* i* e tem como base o léxico estendido da linguagem – LEL.

Para explicar as etapas do método ERi*c, esta dissertação irá utilizar o exemplo do sistema de Controle de Caixa de Restaurante, também presente em

[2]. A Figura 3.2 apresenta um esquema simplificado do método, que está dividido em seis etapas.

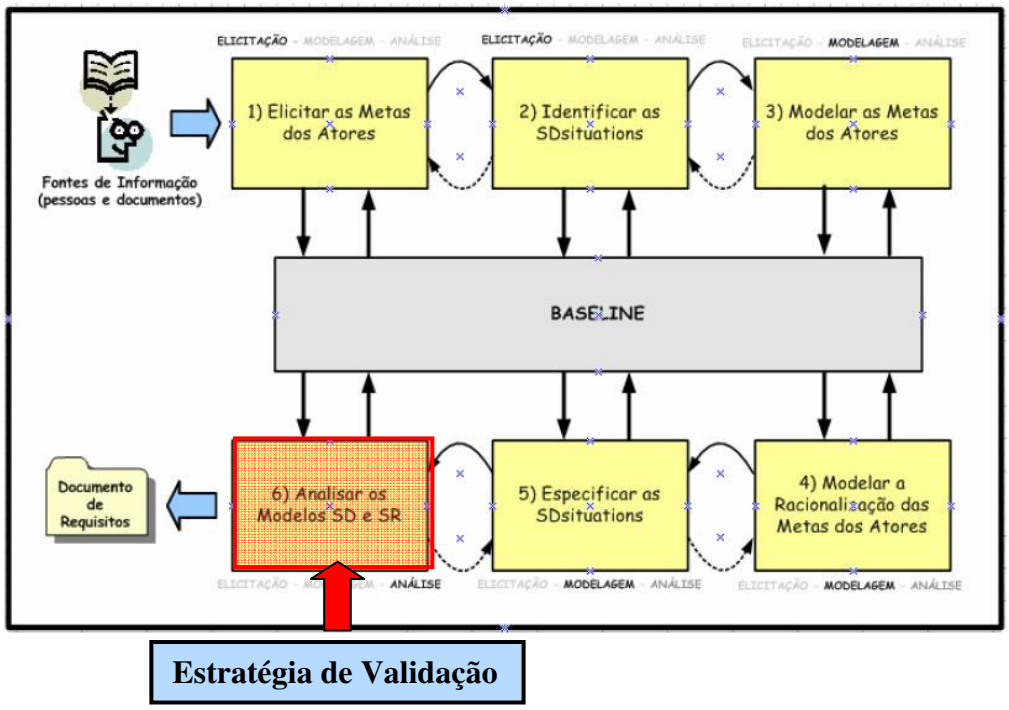

Figura 3.2 – Etapas do método ERi*c. Adaptado de [2].

A Figura 3.2 apresenta as seis etapas do método, que serão seguidas neste trabalho. A primeira etapa, *elicitar metas dos atores*, traz a contribuição da técnica AGFL, fazendo a elicitação a partir do LEL. A segunda etapa, *identificar as SDsituations*, mostra como contribuição as situações de dependência estratégica, definindo blocos de dependência que compartilham intencionalidade situacional. A etapa três, *modelar as metas dos atores*, traz os painéis de intencionalidade, ou diagramas IP, modelando em um diagrama apenas metas concretas e flexíveis e suas relações, simplificando o entendimento dos diagramas SR.

Na quarta etapa do método, *modelar a racionalização das metas dos atores*, ocorre o detalhamento de metas concretas e flexíveis e a derivação de modelos SR a partir de modelos SD. A quinta etapa, *especificar as SDsituations*, especifica as situações de dependência estratégica através da técnica de cenários [13], [17]. Esta etapa do método é bastante importante, pois a descrição das *SDsituations* em cenários servirá como entrada da ferramenta de simulação utilizada neste trabalho. Na sexta e última etapa, *analisar modelos SD e SR*, ocorre a verificação dos

modelos SD e SR através do i* Diagnoses. Nosso trabalho propõe uma mudança para melhoria da sexta etapa do método ERi*c (Figura 3.2), inserindo nesta etapa uma estratégia baseada em simulação para validar os modelos construídos, sem que os interessados precisem absorver a complexidade dos mesmos.

3.2.1.Elicitar as Metas dos Atores

A primeira etapa do método divide–se em três atividades:

1. Preparar o LEL – léxico estendido da linguagem
2. Definir AGFL – metas dos agentes vindas do léxico
3. Refinar as metas

3.2.1.1. Preparar o Léxico Estendido da Linguagem

Para construir o LEL, o engenheiro de requisitos deve inicialmente identificar todas as fontes disponíveis (pessoas e documentos) que forneçam informações sobre o Universo de Informações (UdI)[3]. É importante que o engenheiro de requisitos siga os passos recomendados em [13] para a construção do léxico. A tabela a seguir, presente em [18], exibe as regras gerais para a definição de símbolos.

Tabela 3.1 – Regras gerais para definição de símbolos [18].

Noção		Impacto
Sujeito	Quem é o sujeito	Quais ações ele executa
Verbo	Quem realiza, quando acontece e quais os procedimentos envolvidos	Quais os reflexos da ação no ambiente (outras ações que devem ocorrer) e quais os novos estados decorrentes
Objeto	Definir o objeto e identificar outros objetos com os quais se relaciona	Ações que podem ser aplicadas ao objeto
Estado	O que significa e quais ações levaram a este estado	Identificar outros estados e ações que podem ocorrer a partir do estado que se descreve

[3] O Universo de Informações (ou Universo de Discurso – UofD) inclui todas as fontes de informação e todas as pessoas relacionadas ao software.

Para melhorar o entendimento, a Figura seguinte apresenta alguns exemplos do léxico do sistema de controle de caixa de restaurante, elicitados por Franco em [19].

Nome:	cliente
Noção:	1 - Pessoa que consome **opção** do **restaurante**.
Classificação:	sujeito
Impacto(s):	1 - **senta a mesa**, faz **pedido**, troca de **mesa**, pede **conta**, paga **conta**.
Sinônimo(s):	fregues, clientes.

Nome:	mesa
Noção:	1 - Um dos componentes do **restaurante**. 2 - Cada mesa tem um **num. da mesa**.
Classificação:	objeto
Impacto(s):	1 - O **cliente** pode **sentar a mesa**. 2 - O **cliente** pode **trocar de mesa**. 3 - A **mesa está aberta** ou **mesa está fechada**.
Sinônimo(s):	mesas..

Nome:	calcular o total
Noção:	1 - tarefa realizada pelo **caixa**. 2 - acontece quando o **caixa fecha a mesa**. 3 - para cada **comanda**, o **caixa** verifica o **preço** dos pedidos e adiciona a **soma**. 4 - se o **restaurante** cobra **10%** , o **caixa** calcula os **10%** e adiciona a **soma**.
Classificação:	verbo
Impacto(s):	1 - o **caixa** tem a nota.
Sinônimo(s):	calcula o total,, calcula a soma,.

Figura 3.3 – Exemplo de símbolos do LEL [19].

3.2.1.2. Definir Metas dos Agentes Vindas do Léxico – AGFL

O ponto central desta atividade é descobrir a motivação (o porquê) que está por trás de cada ação. Segundo [2], quando uma ação muda de estado, esta ação define metas concretas. Quando uma ação fornece uma "qualidade" a um estado, esta ação define metas flexíveis. Nesta atividade devemos identificar os atores e extrair as metas relativas aos atores a partir dos impactos dos símbolos pertencentes ao LEL.

No LEL, os símbolos do tipo sujeito e os que praticam ações sobre objetos são bons candidatos a atores. Considerando o exemplo do sistema de Controle de Caixa de Restaurante, o léxico indica três atores: *cliente*, *garçom* e *caixa*.

Uma vez identificados os atores, devemos passar ao passo seguinte, de capturar as metas dos atores a partir do LEL. Oliveira [2] sugere o uso de *templates* diferentes para símbolos do tipo sujeito, objeto, verbo e estado, levando também em consideração metas concretas e metas flexíveis. Para definir as metas, o engenheiro de requisitos deve responder o porquê.

A Figura 3.4 exibe alguns exemplos dos *templates* preenchidos, para ações concretas e flexíveis. Uma ação concreta não pode ser descrita por verbos pouco precisos (controlar, avaliar, apurar, etc.). Já ações flexíveis são pouco concretas e não se pode identificar um resultado concreto a princípio.

É importante observar nas Figuras 3.4 e 3.5 a presença de cada impacto dos símbolos e a motivação envolvida na ação (porquê).

TIPO: SUJEITO	<meta concreta>				ATOR
-- impacto resposta ao porquê?	<sujeito/objeto LAL>	seja/ esteja	<verbo>		<sujeito LAL>
CAIXA					
-- Abre mesa					
Porque **caixa** deseja que	cliente	seja	servido	por	**garçom**
Porque **garçom** deseja que	pedido	seja	feito	por	**cliente**
Porque **caixa** deseja que	mesa	esteja	faturando		
-- Bota a comanda no escaninho					
Porque **caixa** deseja que	comanda	seja	alocada		
-- Fecha mesa					
Porque **caixa** deseja que	mesa	seja	fechada		
-- Transfere de mesa para mesa					
Porque **cliente** deseja que	mesa	seja	trocada	por	**garçom**
Porque **garçom** deseja que	cliente	seja	servido		
Porque **cliente** deseja que	pedido	seja	servido	por	**garçom**
Porque **garçom** deseja que	pedido	seja	feito	por	**cliente**
-- Retira as comandas do escaninho					
Porque **caixa** deseja que	mesa	seja	fechada		
-- Calcula total					
Porque **caixa** deseja que	comandas	sejam	somadas		
**-- Confere pagamento					
Conferir → ação flexível					

MESA					
-- O **cliente** pode sentar a mesa.					
Para que	opção	seja	servida	por	**garçom**
-- O **cliente** pode trocar de mesa.					
Para que	cliente	seja	servido	por	**garçom**
-- A mesa está aberta ou está fechada.					
Para que	mesa	seja	servida	por	**garçom**

Figura 3.4 – Exemplo de *template* para ações concretas (sujeito e objeto) [2].

A Figura 3.4 mostra a definição de metas concretas para cada impacto que menciona uma meta concreta. No caso de um impacto mencionar uma ação flexível, como podemos ver no impacto caixa → confere pagamento sendo tratado como uma ação flexível (conferir), presente no *template* da Figura 3.5.

TIPO: AÇÃO FLEXÍVEL	<meta-flexível>			
-- impacto resposta ao porquê?	<tipo atrib. qualidade	[topico]> sujeito/objeto LAL	< meta associada >	< ator >
CAIXA				
-- confere **pagamento**.				
Porque	correta	[conta]	sem erro [pagamento]	**caixa**
CONTA				
-- É conferido pelo **caixa**.				
Porque	correta	[conta]	sem erro [pagamento]	**caixa**
NUM. DA MESA				
-- O **caixa** verifica o num. da **mesa** da **comanda**.				
Porque	corretas	[comandas]	comandas estejam organizadas	**caixa**

Figura 3.5 – Exemplo de *template* para ações flexíveis (sujeito e objeto) [2].

Em [2], Oliveira explica detalhadamente como preencher os *templates* dos símbolos sujeito, verbo, objeto e estado, trabalhando os aspectos de ações concretas e metas flexíveis e fornecendo algumas heurísticas para definição de metas.

3.2.1.3. Refinar as Metas

Após definir as metas dos agentes vindas do léxico, torna–se necessário organizá–las para facilitar o entendimento das mesmas, além de facilitar a identificação de situações de dependência.

O objetivo desta atividade é agrupar as metas (concretas e flexíveis) por ator e organizá–las em ordem cronológica. A Figura 3.6 mostra um exemplo de refino das metas concretas e flexíveis agrupadas por ator, com repetições excluídas e ordenadas cronologicamente, conforme [2].

caixa					
	mesa	seja	aberta		
	mesa	seja	servida	por	**garçom**
Rentável [mesa]	mesa	esteja	faturando		
	comanda	seja	alocada		
Corretas [comandas]	comandas	estejam	organizadas		
	comandas	sejam	somadas		
	10 %	seja	calculado		
	total	seja	calculado		
	controle de caixa	seja	realizado		
Corretas [comandas] Sem erro [pagamento]	mesa	seja	cobrada	por	**garçom**
Honesto [pagamento] Rentável [mesa]	conta	seja	paga	por	**cliente**
	controle interno	seja	realizado		
Sem erro [pagamento] Correta [conta]	mesa	seja	fechada		
	10 %	seja	pago	por	**cliente**
	garçom	seja	remunerado		

Figura 3.6 – Metas agrupadas por ator cronologicamente organizadas [2].

3.2.2.Identificar as Situações de Dependência Estratégica

Esta etapa é composta por três atividades:

1. Distinguir *SDsituations*
2. Reconhecer interdependências entre *SDsituations*
3. Construir diagrama de *SDsituations*

3.2.2.1. Distinguir *SDsituations*

Nesta atividade, o engenheiro de requisitos deve perceber como as metas compõem conjuntos para formar situações de negócio. Os atores possuem metas (que foram previamente elicitadas na etapa anterior), e essas metas são fortemente ligadas entre si de modo a formar uma situação estratégica de cooperação entre os atores, ou seja, uma *SDsituation* [7].

No caso do sistema de Controle de Caixa do Restaurante, foram identificadas quatro *SDsituations* [2]:

1. Liberação da mesa [caixa e garçom]
2. Atendimento da mesa [cliente, caixa e garçom]
3. Fechamento da conta [cliente, caixa e garçom]
4. Rateio dos 10% [caixa e garçom]

Podemos observar que o nome das *SDsituations* obedecem a sintaxe proposta em [2]: **<substantivo da situação + objeto>**. A Figura 3.7 mostra as metas de um dos atores (caixa) organizadas por *SDsituations*.

caixa						
	1	mesa	seja	aberta		
	2	mesa	seja	servida	por	garçom
	2	comanda	seja	alocada		
Corretas [comandas]	2	comandas	estejam	organizadas		
	3	comandas	sejam	somadas		
	3	10 %	seja	calculado		
	3	controle de caixa	seja	realizado		
Correta [conta]	3	mesa	seja	cobrada	por	garçom
Sem erro [pagamento]	3	conta	seja	paga	por	cliente
	3	mesa	seja	fechada		
	4	controle interno	seja	realizado		
	4	10 %	seja	pago	por	cliente
	4	garçom	seja	remunerado		
Rentável [mesa]	4	mesa	esteja	faturando		

Figura 3.7 – Metas concretas e flexíveis de *caixa* organizadas por *SDsituations*.

3.2.2.2. Reconhecer Interdependências entre *SDsituations*

Neste caso, o engenheiro de requisitos deverá observar as *SDsituations* e determinar suas interdependências (física, lógica ou temporal). Em outras palavras, esta atividade serve para determinar como as situações estão relacionadas dentro do processo de negócio.

3.2.2.3. Construir Diagramas de *SDsituations*

Esta atividade propõe a representação gráfica de todas as situações de dependência estratégica em um único diagrama [7]. É importante mostrar a representação do fator tempo e as interdependências.

A Figura 3.8 mostra o diagrama de *SDsituations* para o sistema de Controle de Caixa de Restaurante [2].

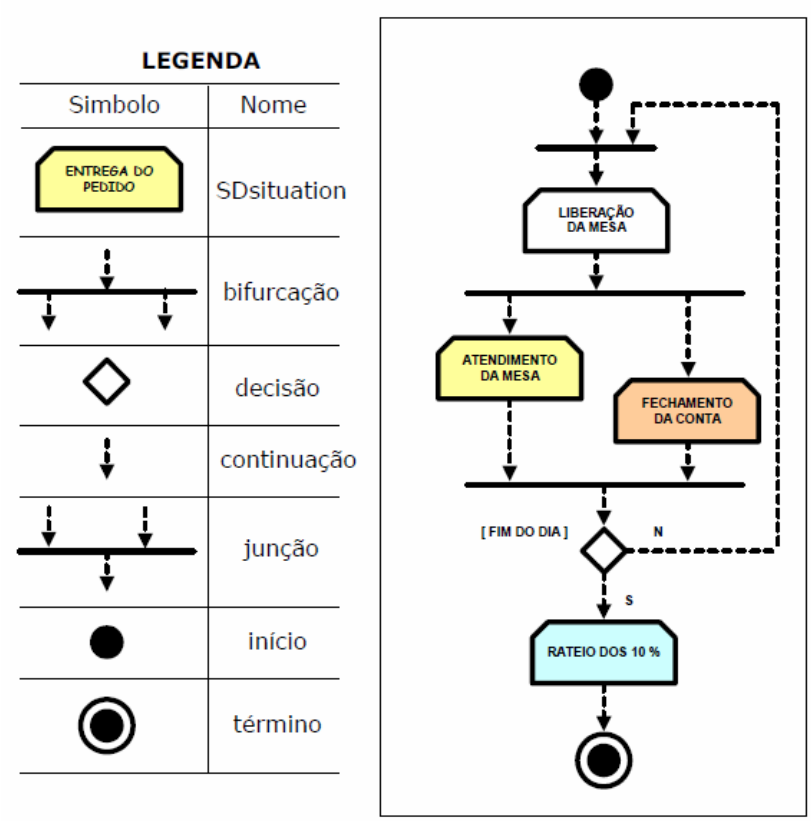

Figura 3.8 – Diagrama de *SDsituations* – Controle de Caixa de Restaurante [2].

3.2.3.Modelar as Metas dos Atores

A terceira etapa do método ERi*c possui duas atividades:

1. Identificar agentes, posições e papéis
2. Criar os painéis de intencionalidade

3.2.3.1. Identificar Agentes, Posições e Papéis

Em cada *SDsituation*, atores trabalham e colaboram para atingir a meta da situação determinada, podendo para isso, assumir posições e desempenhar papéis. Nesta atividade deve–se tomar como base as metas elicitadas de cada ator dentro de cada situação estratégica e observar se existem posições ou papéis que podem ser identificados. Essa tarefa pode ser desempenhada realizando o preenchimento de uma tabela ou mapeando graficamente através de um diagrama SA (*strategic actor*). A Figura 3.9 apresenta o diagrama SA para o Controle de Caixa de Restaurante.

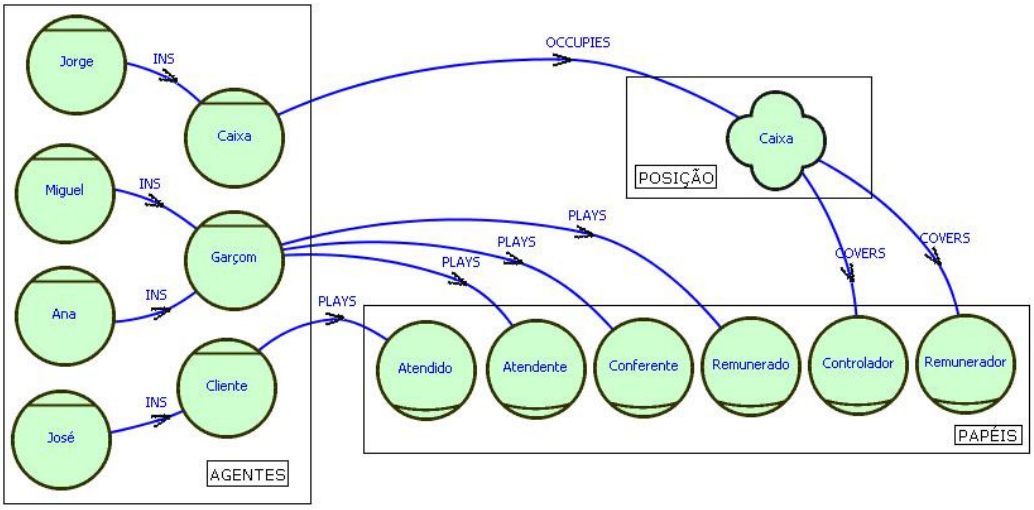

Figura 3.9 – Diagrama SA – Controle do Caixa de Restaurante

3.2.3.2. Criar os Painéis de Intencionalidade

Nesta atividade, deve–se preparar um diagrama IP para cada *SDsituation*. Para cada ator em um diagrama IP, deve–se decidir sobre a ordenação das metas no eixo do ator, de baixo para cima, de modo que as metas iniciais fiquem na parte de baixo do eixo.

É importante trabalhar com um ator de cada vez, representando primeiramente as relações de contribuição (entre metas flexíveis), seguidas das correlações (entre metas flexíveis e metas concretas) e por fim, deve–se representar as relações de dependência entre as metas concretas dos atores.

A Figura 3.10, retirada de [2] exemplifica um diagrama IP.

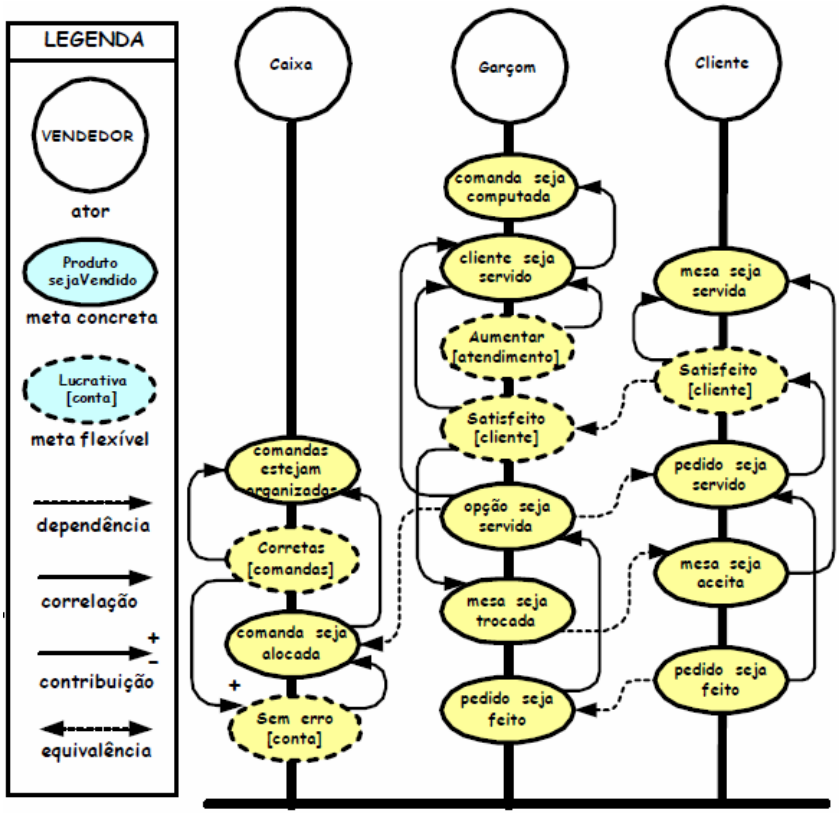

Figura 3.10 – Diagrama IP – Atendimento da Mesa [2].

Os diagramas IP são importantes no entendimento para o próximo passo. Por ser um diagrama mais simples, possuindo apenas atores, metas (concretas e

flexíveis) e suas relações, serve como intermediário no entendimento dos diagramas SD e SR, possuidores de maior complexidade.

3.2.4.Modelar a Racionalização das Metas dos Atores

A quarta etapa é formada por duas atividades:

1. Construir Modelos SD
2. Construir Modelos SR

3.2.4.1.Construir Modelos SD

Para executar esta atividade, o engenheiro de requisitos deve ter em mãos o diagrama de *SDsituations* e os diagramas IP. O objetivo desta atividade é definir as dependências estratégicas entre os atores. Existem quatro tipos de dependências (por meta, tarefa, recurso ou meta flexível). Cabe ao engenheiro de requisitos decidir qual dependência é mais indicada para cada caso. A Tabela 3.2 mostra cada tipo de dependência de acordo com a relação entre os atores e o grau de liberdade associado.

Tabela 3.2 – Regras para definição de dependências estratégicas.

Dependência	Relação de Colaboração entre Atores	Grau de Liberdade
Meta concreta	*dependee* tem total liberdade para tomar decisões a fim de satisfazer a meta	pleno
Tarefa	*dependee* obedece critérios bem estabelecidos pelo *depender*	parcial
Recurso	*dependee* disponibiliza o recurso desejado pelo depender	inexistente
Meta flexível	o *depender* não influencia o *dependee*, e tem a decisão final de aceitar ou não	pleno mas avaliável

Baseado na Tabela 3.2, o engenheiro de requisitos deve elaborar um modelo SD para cada *SDsituation*. As dependências estratégicas estão presentes no diagrama IP, feito na etapa anterior.

Sendo assim, voltando ao diagrama IP da Figura 3.10, relativo à *SDsituation Atendimento da Mesa,* pode–se observar uma dependência estratégica entre os atores *caixa* e *garçom* e quatro dependências entre os atores *garçom* e *cliente*.

Em seguida, o engenheiro de requisitos deve analisar o grau de liberdade que essas dependências terão no contexto organizacional, a fim de expressar o tipo de dependência que estará presente no modelo SD.

Fazendo as análises necessárias considerando o diagrama IP supracitado, obteve–se o seguinte modelo SD para *Atendimento da Mesa*:

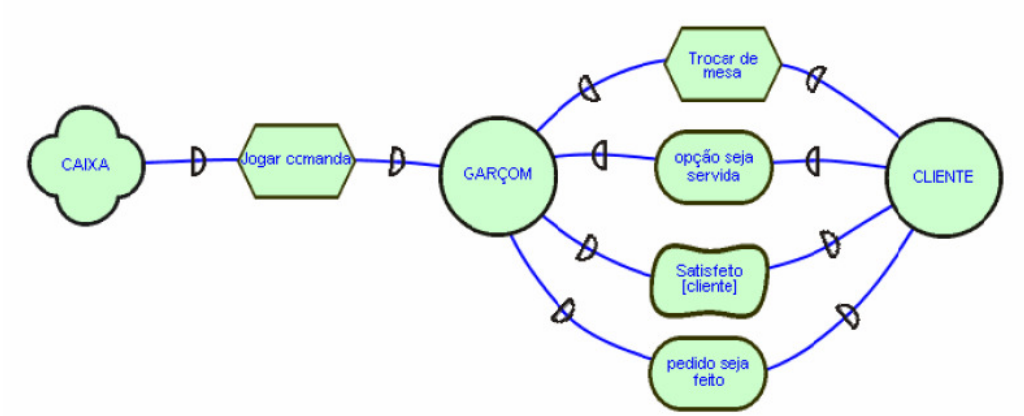

Figura 3.11 – Modelo SD para Atendimento da Mesa [2].

3.2.4.2.Construir Modelos SR

Após a construção do modelo SD, o engenheiro de requisitos poderá construir os modelos SR, utilizando para isso os diagramas IP e diagrama de *SDsituation*.

Cada modelo SR deve ser criado para resolver uma determinada situação estratégica. Os elementos de dependência entre atores (metas, tarefas, recursos ou metas flexíveis) e suas respectivas ligações devem estar representados dentro da linha limite de cada ator, de forma a cumprir sua intencionalidade [1].

As dependências ocorridas no modelo SD também devem estar no modelo SR. Elas devem estar fora da linha limite dos atores, porém unindo elementos internos a cada ator que participa da dependência.

As metas devem ser cumpridas por tarefas ligadas a elas através de elos meio–fim. Tarefas–meio deverão ser ligadas por elos de decomposição. É importante também identificar recursos a serem compartilhados por tarefas.

As metas flexíveis devem ser mapeadas as tarefas as quais elas exprimem qualidade a intencionalidade associada, observando os elos de contribuição com outras metas flexíveis.

Com base nessas regras, o diagrama SR para a situação Atendimento da Mesa, é apresentado na Figura a seguir:

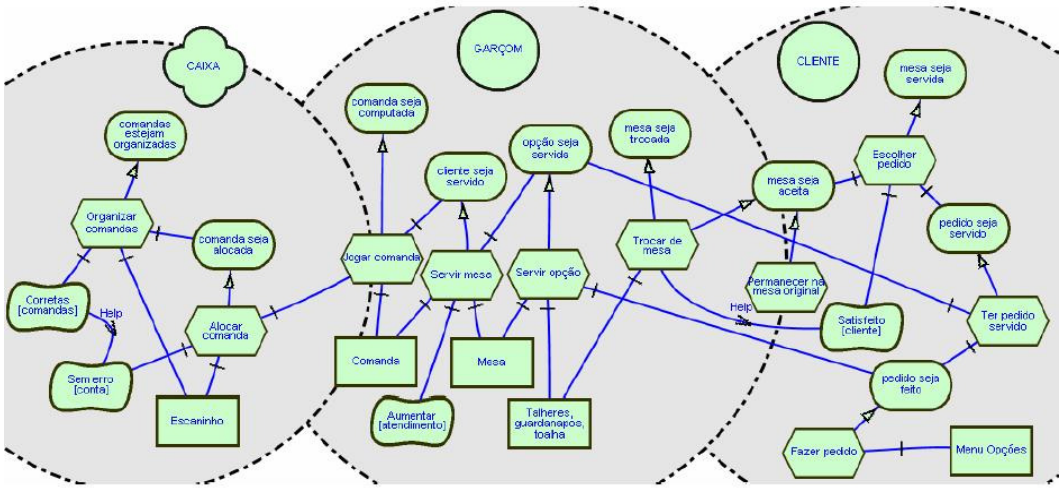

Figura 3.12 – Modelo SR da situação Atendimento de Mesa. Retirado de [2].

3.2.5.Especificar as *SDsituations*

Nesta etapa ocorre a descrição das situações de dependência estratégica em cenários, utilizando os artefatos construídos nas etapas anteriores deste método. Esta etapa é de suma importância para nossa estratégia de validação de modelos i*. A descrição das *SDsituations* em cenários deve representar fielmente a intencionalidade mapeada para os modelos SD e SR. Quanto mais fiel for a descrição, melhor será o modelo obtido após a simulação.

O modelo de cenário adotado neste trabalho é uma estrutura composta pelas entidades: *título, objetivo, contexto, recursos, atores, episódios, exceções* e *restrições.* A Figura 3.13 exibe o modelo de cenário que será utilizado, presente em [13].

Cenário: descrição de uma situação do domínio da aplicação.
Sintaxe: Título + Objetivo + Contexto + $\{Recursos\}_1^N$ + $\{Atores\}_1^N$ + $\{Episódios\}_2^N$ + {Exceções}

Título: identificação do cenário. Em caso de sub-cenário, o título é o mesmo do episódio que o sub-cenário explica, sem as restrições.
Sintaxe: Frase | ([Ator | Recurso] + Verbo + Predicado)

Objetivo: meta que o cenário pretende atingir no domínio da aplicação. O cenário descreve o alcance de um objetivo.
Sintaxe: [Ator | Recurso] + Verbo + Predicado

Contexto: composto por pelo menos um dos seguintes sub-componentes:
Localização Geográfica: local fisco que ocorre o cenário.
Localização Temporal: especificação temporal do desenvolvimento do cenário.
Pré-condição: estado inicial do cenário
Sintaxe: {Localização Geográfica} + {Localização Temporal} + {Pré-condição}
onde Localização Geográfica é
Frase + {Restrição}
onde Localização Temporal é
Frase + {Restrição}
onde Pré-condição
[Sujeito | Ator | Recurso] + Verbo + Predicado + {Restrição}

Recursos: elementos físicos relevantes ou informações que devem estar disponíveis no cenário.
Sintaxe: Nome + {Restrição}

Atores: pessoas, dispositivos ou estruturas organizacionais que tem um papel no cenário.
Sintaxe: Nome

Episódios: conjunto de ações que detalham o cenário e fornecem seu comportamento. Um episódio também pode ser descrito como um cenário.
Sintaxe:
<episódios>::== <séries grupo> | <séries episódios>
<séries grupo>::== <grupo> <grupo> | <grupo não sequencial > | <séries grupo> <grupo>
<grupo>::==<grupo sequencial> | <grupo não sequencial>
<grupo sequencial>::==<sentença básica> | <grupo sequencial> <sentença básica>
<grupo não sequencial>::== #<séries episódio>#
<séries episódio>::== <sentença básica> <sentença básica> |
<séries episódio> <sentença básica>
<sentença básica>::== <sentença simples> | <sentença condicional> | <sentença optativa>
<sentença simples>::== <sentença episódio>
<sentença condicional>::== SE <condição> ENTÃO <sentença episódio>
<sentença optativa>::== [<sentença episódio>]
onde <sentença episódio> é descrita como:
(([Ator | Recurso] + Verbo + Predicado) | ([Ator | Recurso] + [Verbo] + Título)) + {Restrição}

Exceções: geralmente refletem a falta ou mau funcionamento de um recurso necessário. O tratamento de uma exceção pode ser expresso por outro cenário.
Sintaxe: Causa [(Solução)]
onde Causa é:
Frase | ([Sujeito | Ator | Recurso] + Verbo + Predicado)
onde Solução é:
Título

Restrições: requisito de qualidade ou escopo que se refere a uma dada entidade. É um atributo dos recursos, episódios básicos ou sub-componentes do contexto.
Sintaxe:
([Sujeito] | Ator | Recurso] + [Não]Deve(m) + Verbo + Predicado) | Frase

"+" significa composição, {x} significa zero ou mais ocorrências de x, ( ) é usado para agrupamento, "|" é usado como OU e [x] denota que x é opcional.

Figura 3.13 – Modelo de Cenário. Traduzido de [13].

Durante a descrição das situações de dependência estratégica, utilizando o modelo de cenário apresentado na Figura 3.13, é imperativo: maximizar o uso de símbolos do LEL, dar atenção especial aos atores que participem da interação, colocando–os de maneira explícita, e representar os atributos de qualidade com a mesma sintaxe das metas flexíveis.

Os Títulos das *SDsituations* já foram obtidos anteriormente, na etapa 3.2.2.1 (Distinguir as *SDsituations*). O campo Objetivo não tem a obrigatoriedade de estar dentro da situação. De acordo com [2], o objetivo **está além** da *SDsituation*, sendo considerado assim um estado que os atores desejam alcançar através da situação de dependência. Portanto, o engenheiro de requisitos deve analisar o desejo dos atores dentro da situação de dependência estratégica para poder determinar o Objetivo da mesma.

O campo Contexto é composto por Localização Geográfica (local físico que ocorre o cenário), Localização Temporal (especificação temporal do desenvolvimento do cenário) e Pré–condição (estado inicial do cenário). Estes campos devem ser preenchidos pelo engenheiro de requisitos de acordo com seu entendimento da situação, procurando sempre utilizar a maior quantidade possível de símbolos do léxico.

Em Recursos, o engenheiro de requisitos deve inserir elementos físicos que devem ser disponibilizados no desenvolvimento do cenário. Os Atores já foram determinados anteriormente, durante a etapa 3.2.1.2 (Definir Metas dos Agentes Vindas do Léxico – AGFL).

Os Episódios dos cenários devem refletir a intencionalidade dos atores participantes da situação de dependência estratégica. Para isso, o engenheiro de requisitos deve tomar como base os artefatos produzidos nas etapas anteriores (painéis IP, diagramas de *SDsituations* e diagramas SR). Para descrever os episódios, deve–se perceber como as metas, tarefas, recursos e metas flexíveis se encaixam dentro cada situação, observando principalmente a **ordem** em que cada evento ocorre na *SDsituation*. O mais importante neste ponto é deixar presente a intencionalidade de cada ator ao tentar satisfazer suas metas.

O campo Exceções é destinado a registrar o tratamento de alguma falta ou situação excepcional ocorrida durante o desenvolvimento normal dos episódios. É preenchido de acordo com o entendimento do domínio.

Por fim, o campo Restrição é destinado a representar duas situações: alguma imposição que restrinja um episódio (restrições comuns) ou requisitos de qualidade (metas flexíveis). As metas flexíveis já foram identificadas pelo engenheiro de requisitos em etapas anteriores do método (3.2.1.2 Definir Metas dos Agentes Vindas do Léxico – AGFL). O tipo de tratamento utilizado em cada caso (restrições comuns ou metas flexíveis) será descrito durante a atividade de Transformação, na seção 3.3.

A Figura a seguir demonstra as regras para especificação de *SDsituations* supracitadas através do cenário Atendimento da Mesa. É importante observar que os símbolos sublinhados são pertencentes ao LEL.

Título: ATENDIMENTO DA MESA
Objetivo: **Restaurante possa faturar**
Contexto:
Localização geográfica: Restaurante
Localização temporal: Durante o atendimento da mesa.
Precondição: Mesa está aberta.
Recursos: comanda em branco, mesa, menu opções e escaninho
Atores: Cliente e garçom
Episódios: Cliente faz pedido.
Cliente pergunta sobre opção.
Garçom responde sobre opção.
Cliente escolhe opção.
Garçom atende a mesa.
Garçom anota pedido.
Garçom joga a comanda.
Restrição: Para transferir de mesa para mesa, a mesa de destino precisa estar fechada.
Exceções: Se cliente desejar trocar de mesa:
(Caixa deve transferir de mesa para mesa)
Metas flexíveis: Satisfeito [cliente], Corretas[comandas], Aumento[atendimento]

Figura 3.14 – Exemplo de representação em cenário.

3.2.6.Analisar os Modelos SD e SR

Nesta etapa será empregada a validação baseada em simulação, objeto principal deste trabalho. Seguindo os passos do método ERi*c mostrados até aqui, o engenheiro de requisitos estará apto a obter modelos bem construídos, que deverão ser validados com os interessados.

As próximas seções irão tratar do processo de transformação dos cenários para a entrada da ferramenta de simulação *UCEd*, e em seguida serão apresentadas

as etapas para colher o *feedback* dos interessados e as heurísticas que irão tratar os impactos do *feedback* (erros, omissões e/ou discrepâncias) nos modelos construídos.

3.3. Transformar

Com os modelos i* em mãos, o engenheiro de requisitos deve trabalhar na transformação dos cenários construídos, para que os mesmos possam servir de entrada para a ferramenta *UCEd*.

3.3.1.Heurísticas de Transformação

A ferramenta de simulação *UCEd* não possui todos os campos presentes no modelo de cenários utilizado para descrever das *SDsituations* neste trabalho (Figura 3.13). Sendo assim, torna–se necessário elaborar algumas heurísticas de transformação para que nenhum conteúdo dos cenários seja perdido durante a simulação. A seguir, apresentam–se as heurísticas por cada elemento de um cenário completo:

- **Título:** presente no modelo e na ferramenta. Mapeamento feito de forma direta.

- **Objetivo:** presente no modelo e na ferramenta. Mapeamento feito de forma direta.

- **Contexto:** o contexto é composto por: Localização Geográfica, Localização Temporal e Pré–condição. Tanto Localização Geográfica quanto Localização Temporal não estão presentes explicitamente no programa de simulação. Deve–se criar no *Domain Model* um *Concept* Localização Geográfica e um *Concept* Localização Temporal, e adicionar

em seus *Possible Values* o valor desejado. Por exemplo, um *Possible Value* para Localização Geográfica pode ser Restaurante [2]. Como sugestão, o engenheiro de requisitos pode explicitar as Localizações Geográficas e Temporais no campo *Description,* para que os mesmos fiquem textualmente destacados no programa.

- **Pré–condição:** está presente no modelo e na ferramenta, sendo seu mapeamento feito de forma direta. Além disso, deve–se inserir também a Localização Geográfica e Temporal, após declará–las como *Concepts*, de forma a mantê–las inseridas no programa. Um exemplo seria a pré–condição do cenário "Atendimento da Mesa" [2]: *Mesa is aberta AND Localização Geográfica is Restaurante AND Localização Temporal is durante o atendimento da mesa.*

- **Recursos:** não estão presentes explicitamente na ferramenta. Quando o símbolo de um recurso aparecer nos episódios, possuir estado e/ou restrições, o mesmo deve ser tratado como um *Concept,* sendo declarado no *Domain Model,* bem como devem estar presentes seus estados e/ou restrições. Por exemplo, o recurso *Mesa* foi declarado como um *Concept e seus* estados, *aberta* e *fechada,* como sendo seus *Possible Values.*

- **Atores:** a ferramenta possui para a declaração dos atores, os campos ***Primary Actor*** e ***Participants***. No primeiro campo, insere–se o ator responsável por iniciar o cenário e que deseja ter seu objetivo alcançado. Os demais atores são inseridos no segundo campo.

- **Episódios:** a ferramenta possui o campo ***Steps*****,** onde os episódios devem ser inseridos. É importante ressaltar que a ferramenta suporta todos os tipos de sentenças (sequencial, não sequencial, condicional e optativa), citados em [13].

- **Exceções:** este campo não está explicitamente presente no programa. Neste caso, as exceções são tratadas como passos alternativos

(***Alternatives***) dentro da ferramenta. O engenheiro de requisitos fica responsável por escolher na ferramenta em qual episódio será inserido o(s) passo(s) alternativo(s).

- **Restrições:** este campo não está presente explicitamente na ferramenta. Neste caso, serão dados tratamentos diferentes a dois grupos distintos de restrições: restrições comuns e restrições do tipo *softgoals* (metas flexíveis) [1], [2], [3]. As restrições comuns são tratadas como passos alternativos, cabendo ao engenheiro de requisitos inseri–las no episódio mais adequado. As restrições do tipo meta flexíveis são mais complexas, necessitado de um tratamento adequado para sua inserção nos cenários da ferramenta. As restrições do tipo metas flexíveis serão tratadas detalhadamente na próxima seção.

Cabe ressaltar que existem campos na interface do programa que não pertencem aos elementos típicos de um cenário: ***System Under Design, Primary Actor, Participants, Invariant, Success Post Condition*** e ***Follow Use Cases*****.** Os campos ***Primary Actor*** e ***Participants*** já foram tratados anteriormente. ***System Under Design*** pode ser preenchido sem problemas, caso seja conveniente. ***Invariant*** é uma condição que deve ser mantida durante todo o cenário. Seu preenchimento fica a cargo do engenheiro de requisitos, caso seja necessário, assim como ***Success Post Condition*****.** O campo importante que em alguns casos pode ser preenchido é o ***Follow Use Cases***, utilizado na ferramenta para ligar os cenários na sequência em que eles ocorrem, de acordo com o diagrama de *SDsituations*.

3.3.2.Tratando Restrições do Tipo Metas Flexíveis

Visando a inserção de restrições do tipo metas flexíveis nos cenários transformados para a ferramenta de simulação *UCEd*, iremos utilizar como base os trabalhos de Cysneiros [14] e Neto [15].

O surgimento de metas flexíveis ocorre de maneira natural, durante a etapa de definição das metas dos agentes vindas do léxico – AGFL do método ERi*c.

Muitas metas flexíveis ficam em grande destaque em determinados domínios, sendo captadas de maneira bem simples.

Uma vez que as metas flexíveis já foram elicitadas (Seção 3.2.1.2), devemos agora representá–las de uma forma mais organizada e propícia para inseri–las nos cenários transformados da ferramenta *UCEd*. Para isso, vamos utilizar o grafo de NFRs, proposto por Chung [5], seguindo a seguinte sistematização:

1. Definir a raiz para cada meta flexível elicitada, como sendo a própria *meta flexível* e o [tópico] como sendo um símbolo do LEL. Por exemplo: *Segurança [artigo], Satisfeito [cliente]*.
2. Decompor a meta flexível em submetas, por tipo ou por tópicos [5].
3. Continuar as decomposições até chegar a um determinado nível de granularidade em que se possa visualizar o que é necessário para a operacionalização da meta flexível.

A Figura a seguir mostra um exemplo de grafo NFR, obtido através da meta flexível Corretas [comandas], desde a raiz até suas operacionalizações.

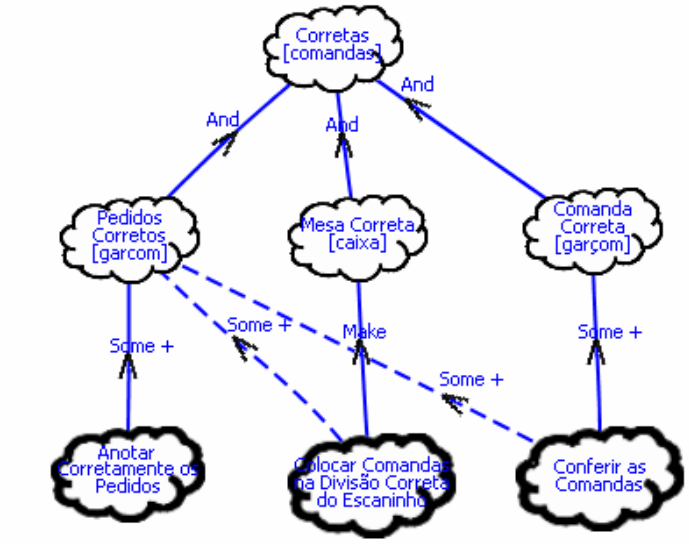

Figura 3.15 – Exemplo de grafo NFR.

Com as metas flexíveis representadas em seus grafos NFRs, o engenheiro de requisitos deve voltar a sua atenção para a solução de conflitos. Ao se visualizar os grafos de maneira relativamente próxima, pode–se verificar

interdependências positivas ou negativas e assim resolver problemas de conflitos que venham a ocorrer entre as metas.

Para que o engenheiro de requisitos não dependa que a organização dos grafos favoreça a identificação de interdependências, [14] propõe as seguintes heurísticas:

1. Comparar todos os grafos de um mesmo tipo. Exemplo: comparar todos os grafos relativos à meta flexível *Desempenho*.
2. Comparar grafos de tipos possivelmente conflitantes. Exemplo: grafos que envolvam *Segurança* e *Desempenho*.
3. Comparar os grafos por pares, de forma a comparar um dado grafo com todos os outros, sem repetir as comparações feitas anteriormente.

A Figura 3.16 [14] mostra um exemplo de metas flexíveis conflitantes. O engenheiro de requisitos deve procurar determinar que desenhos alternativos são possíveis para contornar o conflito ou se uma meta flexível será satisfeita em detrimento às outras.

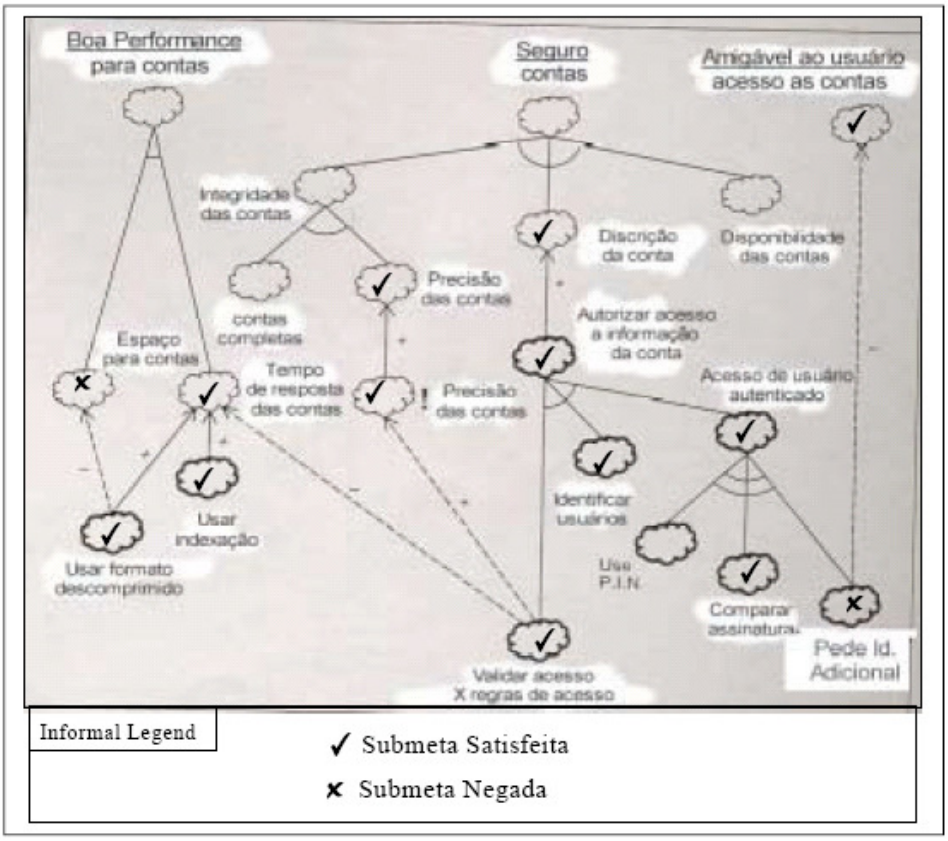

Figura 3.16 – Exemplo de metas flexíveis conflitantes.

A próxima etapa é a integração das metas flexíveis operacionalizadas aos cenários da ferramenta *UCEd*. Para isso, devemos seguir os seguintes passos:

1. Para cada meta flexível elicitada e devidamente representada pelo seu grafo NFR, deve–se avaliar cada uma das suas operacionalizações, e verificar em seu respectivo cenário a existência de episódios que garantam a satisfação dessas operacionalizações.
2. Se os episódios existentes não forem suficientes para garantir a satisfação da meta flexível, o engenheiro de requisitos deve incluir 1 (um) ou mais episódios de forma a satisfazer esta meta flexível. Ao término das inclusões necessárias, deve–se analisar a ocorrência de alguns conflitos que possam vir a ocorrer.
3. Caso um conflito venha a ocorrer, o engenheiro de requisitos deve retornar ao grafo NFR para verificar o quanto prejudicial seria a não satisfação desta meta flexível, e avaliar junto ao interessado possibilidades para contornar o conflito.
4. Ao inserir novos episódios nos cenários do programa *UCEd*, é imperativo que o engenheiro de requisitos coloque após o episódio inserido a seguinte expressão: // *Nome_da_Meta_Flexível* [*Símbolo*]. Essa informação é importante para que a rastreabilidade no modelo seja mantida. As duas barras no início da expressão são para que o programa considere a mesma como um comentário dentro do cenário, evitando a influência do texto durante a compilação deste cenário.

A seguir será exibido um exemplo do cenário *Atendimento da Mesa*, com suas respectivas operacionalizações inseridas nos episódios da ferramenta.

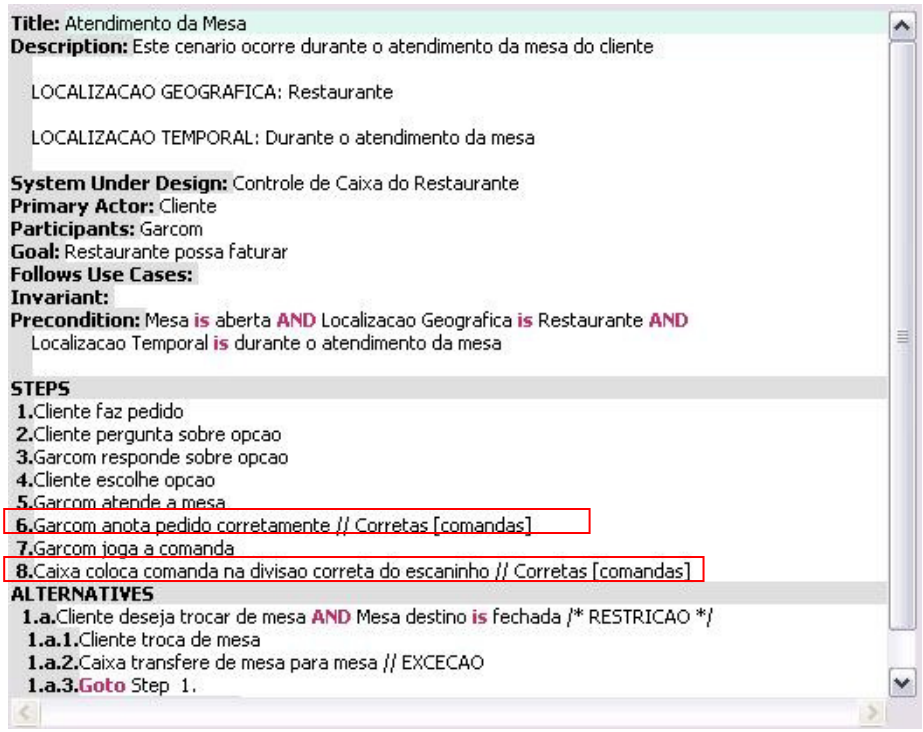

Figura 3.17 – Metas flexíveis inseridas nos cenários da ferramenta.

É valido observar alguns detalhes importantes na Figura 3.17. Como mencionado anteriormente, após inserir novos episódios, é necessário inserir o comentário com a meta flexível que originou o novo episódio (destaque em vermelho), mantendo assim a rastreabilidade do modelo. Esse recurso também é de grande valia para manter a rastreabilidade para exceções e restrições comuns.

Ao término desses passos, o cenário descrito está totalmente adaptado à ferramenta *UCEd*, podendo assim ser simulado e consequentemente validado pelos interessados.

3.4. Validar por Simulação

Nesta etapa ocorre a formalização da validação dos requisitos elicitados. O engenheiro de requisitos deverá preparar a ferramenta para a simulação dos episódios com o interessado. O ambiente em que ocorrerá essa simulação é um ambiente controlado, com registro em áudio dos comentários do próprio

interessado e gravação da tela em que a validação estiver ocorrendo, tudo previamente autorizado por um Termo de Consentimento. Além disso, o engenheiro de requisitos deve estar ao lado do interessado, para coletar dados adicionais e auxiliá–lo na utilização da ferramenta. A seguir, serão explicitados os passos do procedimento de validação.

1. Inserir os cenários transformados na ferramenta de simulação *UCEd*.
2. Preparar o ambiente para a realização da simulação. O ambiente deverá possuir um software de captura de áudio e da tela em que ocorrerá a simulação.
3. Assinatura do Termo de Consentimento pelo interessado.
4. Orientar previamente o interessado quanto aos procedimentos durante a simulação. É importante solicitar que o interessado comente tudo o que achar relevante durante a simulação (dúvidas, discordâncias, erros, entre outras opções). Isso irá tornar a análise posterior mais rica em detalhes.
5. Realizar a validação junto ao interessado. Cada cenário dá origem a uma janela de simulação (Figura 3.18), onde o interessado deverá validar os episódios pertencentes a esses cenários. Neste ponto o engenheiro de requisitos deve estar ao lado do usuário conduzindo a validação.
6. Analisar posteriormente os arquivos resultantes da simulação. Deve-se observar a aplicação das heurísticas de validação (seção 3.4.1), determinando assim os impactos da aplicação da validação por simulação nos modelos i* concebidos inicialmente pelo engenheiro de requisitos.

A Figura 3.18 exibe um exemplo de janela de simulação do programa *UCEd.*

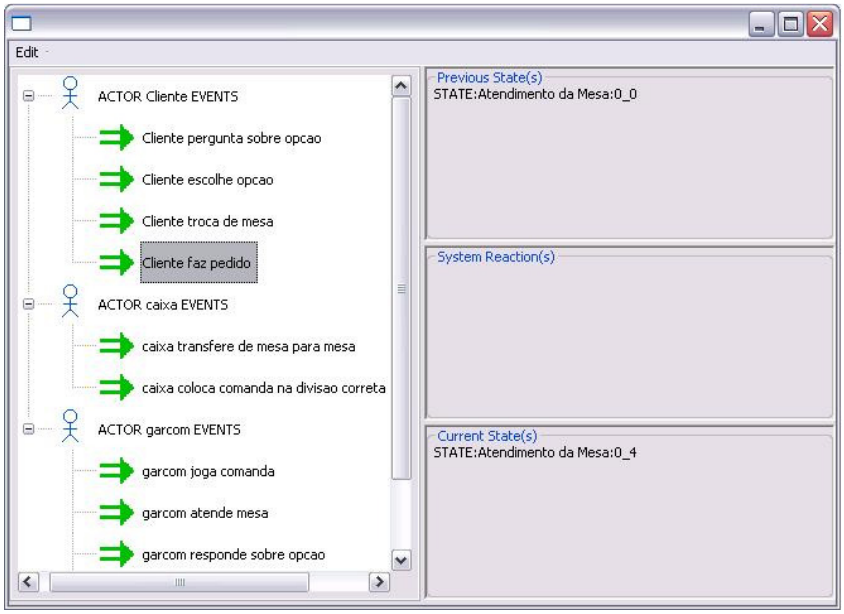

Figura 3.18 – Simulação do cenário Atendimento da Mesa no *UCEd.*

3.4.1.Heurísticas de Validação por Simulação

A validação realizada estará centrada nos seguintes tópicos:

1. Léxico
2. Ordem dos Episódios
3. Completude e Corretude dos Episódios

No caso do léxico, é importante o engenheiro de requisitos observar o surgimento de algum símbolo novo (podendo ser sinônimo ou não), ou se algum símbolo existente teve seu significado alterado.

1. Caso ocorra o surgimento de algum símbolo, o engenheiro de requisitos deve inserir o novo símbolo no LEL (levando em consideração principalmente os impactos deste símbolo), e deve percorrer novamente os passos do método ERi*c até que os impactos do símbolo novo estejam presentes (caso necessário) nos modelos construídos

A ordem inicial dos episódios pode ser alterada pelo interessado. Esse fato pode ser observado pelo engenheiro de requisitos durante a seleção de episódios na ferramenta. Caso seja selecionado um episódio fora da sequência inicial, a ferramenta exibe uma janela chamando a atenção do fato (Figura 3.19), e o engenheiro de requisitos deve se certificar junto ao interessado se ele deseja propositalmente trocar a ordem do episódio selecionado ou não. Cabe considerar que a mera troca de ordem dos episódios não causa impacto direto no léxico.

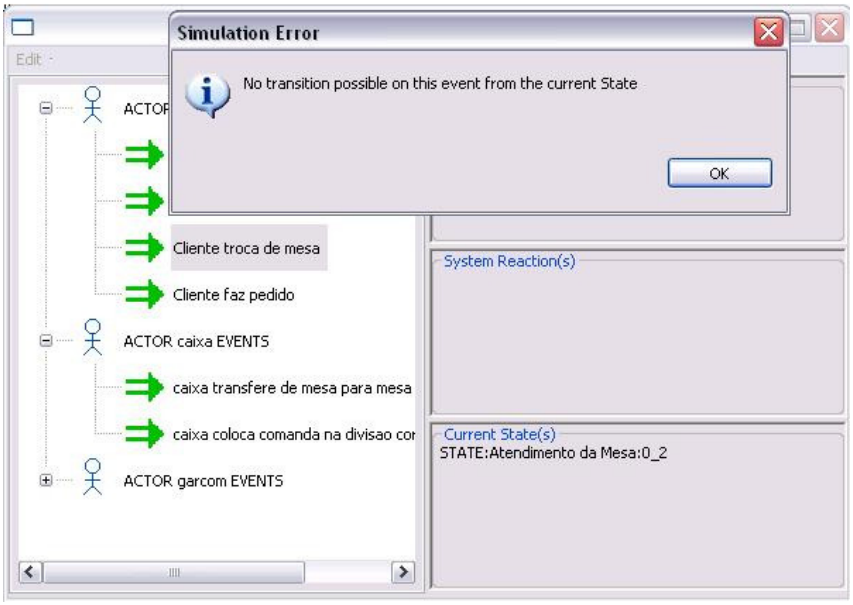

Figura 3.19 – Escolha de um episódio fora da ordem inicial adotada.

2. A mera troca de ordem dos episódios não afeta o léxico, base do método ERi*c. Como os modelos i* possuem caráter atemporal, a troca de ordem dos cenários não deverá impactar os diagramas construídos, desde que essa troca não interfira na intencionalidade dos atores pertencentes a situação de dependência estratégica.

A completude e corretude dos episódios serão observadas pelo engenheiro de requisitos durante a simulação ou após uma minuciosa análise dos registros de áudio e vídeo (do interessado e da tela de simulação), de forma a evitar ao máximo a perda de informações. Ao analisar completude e corretude dos episódios, o engenheiro de requisitos deve ficar atento a 3 (três) casos que podem ocorrer: a criação de novos episódios, a eliminação de episódios e a união de

episódios. Em todos esses casos podem ocorrer impactos no léxico do domínio da aplicação.

3. Em caso de correção dos episódios, o engenheiro de requisitos deve estar atento ao surgimento de novos símbolos para o LEL. Caso ocorra o surgimento de novos símbolos, o engenheiro de requisitos deve seguir is passos da heurística 1 supracitada.

4. Em caso de eliminação de episódios o engenheiro de requisitos deve estar atento se algum símbolo do léxico foi eliminado junto ao episódio (e o mesmo não apareça em nenhum outro lugar). Neste caso, deve–se retirar os impactos deste símbolo dos passos do método ERi*c (obviamente sem perder o rastro desta operação) até que o símbolo retirado deixe de impactar os modelos i* construídos.

5. Em caso de união de episódios, o engenheiro de requisitos deve perceber se foi criado algum símbolo para a união dos episódios, seguindo assim a heurística 1, ou se algum símbolo foi eliminado durante a união, seguindo assim a heurística 4.

Ao término da simulação, o engenheiro de requisitos deverá analisar todo o material procurando discrepâncias, erros e/ou omissões, conforme o foco mostrado nas heurísticas apresentadas anteriormente. Erros e omissões são detectados principalmente durante a simulação com o interessado. Discrepâncias são encontradas principalmente na análise posterior do material. Uma nova simulação deve ser realizada, visando o refinamento dos modelos reconstruídos.

Ao final desta etapa, o engenheiro terá realizado a simulação junto aos interessados, através da ferramenta *UCEd*, e a correção de possíveis alterações encontradas, baseando–se nas heurísticas de validação supracitadas. O capítulo a seguir irá exibir um exemplo da aplicação da estratégia de validação proposta por este trabalho através de um estudo de caso.