LUCAS GONÇALVES CUNHA

FERRAMENTA PARA ELICITAÇÃO DE METAS A PARTIR DO LÉXICO AMPLIADO DA LINGUAGEM

Dissertação apresentada à Universidade Federal de Viçosa como parte das exigências do Programa de Pós-Graduação em Ciência da Computação para obtenção do título de Magister Scientiae.

VIÇOSA
MINAS GERAIS - BRASIL
2014

Ficha catalográfica preparada pela Biblioteca Central da Universidade Federal de Viçosa - Câmpus Viçosa

T
C972
2014

Cunha, Lucas Gonçalves, 1988-
Ferramenta para elicitação de metas a partir do léxico ampliado da linguagem / Lucas Gonçalves Cunha. – Viçosa, MG, 2014.
viii, 92f. : il. (algumas color.) ; 29 cm.

Inclui apêndices.
Orientador: José Luis Braga.
Dissertação (mestrado) - Universidade Federal de Viçosa.
Referências bibliográficas: f.90-92.

1. Engenharia de software. 2. Software - Desenvolvimento. 3. Mineração de dados. 4. IStar. I. Universidade Federal de Viçosa. Departamento de Informática. Programa de Pós-graduação em Ciência da Computação. II. Título.

CDD 22. ed. 005.1

LUCAS GONÇALVES CUNHA

FERRAMENTA PARA ELICITAÇÃO DE METAS A PARTIR DO LÉXICO AMPLIADO DA LINGUAGEM

Dissertação apresentada à Universidade Federal de Viçosa, como parte das exigências do Programa de Pós-Graduação em Ciência da Computação, para obtenção do título de *Magister Scientiae*.

APROVADA: 17 de setembro de 2014.

André Luiz de Castro Leal

Alcione de Paiva Oliveira

Antônio de Pádua Albuquerque Oliveira
(Coorientador)

José Luís Braga
Orientador

Dedico esta dissertação aos meus pais
Albertino e Sueli

Às minhas irmãs
Talitta e Jéssica

Ao meu orientador
José Luis Braga

À minha colega de projeto
Cristiane Aparecida Lana

E aos demais parentes e amigos.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

RESUMO

CUNHA, Lucas Gonçalves, M.Sc., Universidade Federal de Viçosa, setembro de 2014. **FERRAMENTA PARA ELICITAÇÃO DE METAS A PARTIR DO LÉXICO AMPLIADO DA LINGUAGEM**. Orientador: José Luis Braga. Co-Orientador: Antônio de Pádua Albuquerque Oliveira.

A extração de informações sobre um domínio específico é uma grande dificuldade no desenvolvimento de software. Aspectos intrínsecos, linguagem de domínio peculiar e a pluralidade de domínios requerem conhecimento e experiência do Engenheiro de Requisitos para o sucesso na obtenção de informações. Devido à incerteza, falta de padronização e conhecimento sobre o domínio, neste trabalho é apresentada uma abordagem exploratória, baseada em sistemas baseados em conhecimento. Sistemas baseados em conhecimento visam simular situações e soluções utilizando técnicas de inteligência artificial. Neste trabalho é apresentado i*GET, uma ferramenta para auxiliar na elicitação de metas a partir da linguagem do domínio. A i*GET tem como princípio a parametrização de linguagem de domínio contida no Léxico Ampliado da Linguagem, formando o banco de dados. Usando o motor de inferência do CLIPS por meio de regras de produção e do mecanismo de aprendizagem, a ferramenta auxilia o elicitação de metas a serem utilizadas nos modelos construídos com notação do Framework iStar.

ABSTRACT

CUNHA, Lucas Gonçalves, M.Sc., Universidade Federal de Viçosa, September, 2014. **TOOL FOR ELICITATION OF GOALS FROM LANGUAGE EXTENDED LEXICON**. Adviser: José Luis Braga. Co-Adviser: Antônio de Pádua Albuquerque Oliveira.

The information extraction about a particular domain is a big difficulty in software development. Intrinsic aspects, peculiar domain language and the plurality of domains require knowledge and experience from requirements engineer for success in information obtaining. Due to the uncertainty, lack of standards and knowledge about the domain, in this paper is presented an exploratory approach, based on knowledge based systems. Knowledge-based systems aim to simulate situations and solutions using artificial intelligence techniques. In this paper is presented i*GET, a tool for aiding in goals elicitation from the domain language. The i*GET has as principle the domain language parameterization contained in the Language Extended Lexicon (LEL), forming the database. Using CLIPS inference engine through production rules and the learning mechanism, the tool automates and aids the goals elicitation used in models designed with iStar Framework notation.

1 INTRODUÇÃO

Esta seção tem o objetivo de contextualizar o ambiente de desenvolvimento em que se deu a pesquisa realizada neste trabalho. Nas subseções seguintes, são apresentados a motivação, os objetivos e a organização deste trabalho.

1.1 Motivação

A modelagem de sistemas orientados a agentes tem ganhado grande notoriedade nos últimos anos. Diversos estudos têm investido seus esforços para tal abordagem, que já vem sendo desenvolvida há algum tempo (Yu, 1995; Da Silva e De Lucena, 2004). Cada vez mais, espera-se que um software seja robusto, multiplataforma, adaptativo, autônomo e capaz de interagir, comunicar e aprender. Isso ocorre devido à complexidade dos ambientes em que os sistemas estão inseridos, ambientes organizacionais e operacionais altamente mutáveis, onde componentes são alterados, incorporados ou até mesmo excluídos.

As características buscadas nos sistemas vão ao encontro com a caracterização de agente proposta por Yu (Yu, 1995). Segundo Yu, os agentes são caracterizados pela autonomia, habilidade social, comunicação e proatividade, características encontradas em seres humanos. Assim, as propostas de modelagem e construção de softwares tradicionais já não atendem a estas novas demandas. O uso de conceitos como objetivos, metas e ações ao invés de funcionalidades, métodos e procedimento faz com que a modelagem do software seja mais flexível, sendo capaz de lidar com a complexidade intrínseca das aplicações modeladas. O uso desses conceitos busca facilitar a compreensão do ambiente e as interações do software. Devido à ausência de um entendimento correto dos processos e das estruturas da organização, muitos dos sistemas falham no suporte às organizações das quais são partes integrantes.

O uso de metáforas e metas reflete a intencionalidade dos atores organizacionais, o que aproxima do mundo real das organizações em que o software está inserido, facilitando a compreensão daqueles que têm a tarefa de construir um software que vá ao encontro das necessidades organizacionais. O desenvolvimento e

utilização dessas práticas fazem parte de uma nova abordagem a respeito dos requisitos, a Engenharia de Requisitos Orientada a Metas (Goal Oriented Requirement Engineering - GORE). A GORE traz como benefícios a possibilidade de considerar soluções alternativas e potencializar a análise da plenitude dos requisitos, além de fornecer um guia para a rastreabilidade a partir do contexto organizacional (Regev e Wegmann, 2005). Como exemplos de abordagem que se enquadram dentro da proposta da GORE temos TROPOS (Castro et al., 2002), NFR Framework (Chung et al., 2000), KAOS (Van Lamsweerde, 2001), ERi*c (Oliveira, 2008) e o Framework i* (iStar) (Yu, 1995).

Neste contexto, o presente trabalho investiga a questão do desenvolvimento de sistemas com orientação a metas. Mais especificamente propõe uma ferramenta que possa auxiliar do Engenheiro de Requisitos (ER), nas fases de requisitos iniciais, na tarefa de elicitação de metas no processo de criação de modelos de software que utilizam a abordagem GORE.

1.2 Objetivos

O desenvolvimento de software baseado no paradigma de orientação a metas está estreitamente relacionado à intencionalidade. O envolvimento do ER no contexto organizacional é fundamental para entender e extrair requisitos necessários. Segundo Leite (Leite e Oliveira, 1995) as atividades de elicitação da intencionalidade, modelagem dos requisitos e a análise dos modelos fazem parte do processo de requisitos. Assim, as fases que contemplam todas essas atividades são consideradas críticas no desenvolvimento dos sistemas. Isso porque, para construção de um software com qualidade, é preciso que as considerações técnicas estejam em equilíbrio com as sociais e organizacionais desde o início de todo esse processo (Bresciani et al., 2004).

Neste contexto observa-se uma grande lacuna na abordagem de desenvolvimento de sistemas orientada a metas. Todas as abordagens citadas com exemplos são focadas apenas na modelagem (Oliveira 08a). Nos métodos apresentados toda a elicitação fica a cargo da competência do ER. O Framework KAOS (Van Lamsweerde, 2001) sugere a utilização da documentação compostas por palavras que representem intencionalidade, como “objetivo”, “finalidade”, “intenção”, etc. No caso do Framework i* (Yu, 1995), lida apenas com a modelagem, não se preocupando com o restante do processo. O único dos métodos apresentados que aborda a elicitação das metas nos seus processos é o método ERi*c. Apesar de o método contemplar as fases

iniciais, ele não é suportado por nenhuma ferramenta. Além disso, não existe mecanismo para geração de conhecimento que possa auxiliar o ER em elicitações posteriores, fornecendo sugestões para as possíveis metas a serem extraídas ou identificando potenciais metas.

Com o apresentado o objetivo geral do trabalho proposto é a apresentação de uma estratégia e a construção de uma ferramenta que acompanhe o ER na extração de metas, ao apoiar e suportar o mesmo nas etapas inicias de criação de modelos que utilizam a notação de metas para a modelagem de softwares.

Especificamente, pretende-se:

a) apresentar um método baseado no conceito de intencionalidade usado na elaboração de modelos i* (método ERi*c);

b) melhorar a qualidade final das metas elicitadas pelo método ERi*c;

c) facilitar o processo de extração de conhecimento de um domínio representado por meio do Léxico Ampliado da Linguagem (LAL);

d) gerar um banco de conhecimento que possua ações comuns que remetam a metas usadas na construção dos modelos.

1.3 Trabalhos correlatos

O primeiro trabalho, de Cysneiro e Leite (Cysneiros e Leite, 2001), utiliza descrições do LAL para a elicitação de Requisitos Não Funcionais (RNF). A abordagem citada tem como base a ferramenta OORF (Neto, 2000). Essa ferramenta possui uma base de conhecimento que contém um conjunto de RNF e formas comuns de operacionalização para decomposição destes RNF, além de RNF que contribuem e conflitam com um determinado requisito. O processo de extração consiste em uma busca exaustiva guiada pelo ER por meio de descrições elicitadas a partir dos símbolos do LAL. Dessa forma, são adicionados RNF às descrições de noção dos símbolos do LAL. Diferentemente do trabalho de Cysneiro e Leite (Cysneiros e Leite, 2001) e de Neto (Neto, 2000), o nosso trabalho utiliza-se da abordagem de modelos intencionais, não utilizando os conceitos como requisitos funcionais ou requisitos não funcionais e sim metas concretas e flexíveis (softgoals).

O trabalho de Baia et al. (Baía et al., 2012) utiliza a estratégia de investigação do vocabulário do domínio descrito por meio do LAL em busca de sinônimos e características que apontam requisitos de transparências nas descrições do domínio. Nos dois trabalhos, o conhecimento descrito por meio do LAL é transformado e investigado pelas regras de inferência. A distinção entre os trabalhos é o objetivo final da investigação. Em Baia (Baía et al., 2012), o objetivo da averiguação é encontrar requisitos de transparência nos modelos i*, enquanto em nosso trabalho, o objetivo é buscar por metas para a construção dos modelos que serão representados em notação i*.

1.4 Estrutura do Trabalho

O texto da dissertação está organizado da seguinte forma:

O capítulo 2 apresenta os conceitos básicos e as características envolvidas nesse trabalho como: Framework i*, Léxico Ampliado da Linguagem, CLIPS, Sistemas Especialista e o método ERi*c.

O capítulo 3 apresenta a descrição da estratégia e o método utilizado na construção da ferramenta.

O capítulo 4 apresenta um estudo de caso utilizando a abordagem descrita neste trabalho.

O capitulo 5 apresenta as conclusões e contribuições, além de sugestões para trabalhos futuros.

E por fim, se encontram os anexos, referências importantes utilizadas no desenvolvimento dessa dissertação.

2 REFERENCIAL TEÓRICO

Nesta seção, serão apresentados alguns conceitos, métodos e ferramentas que são bases do trabalho. Primeiramente, será apresentado o conceito do LAL (Léxico Ampliado da Linguagem). Em sequência, é dada uma visão geral a respeito do Framework iStar, seus modelos de Razão Estratégica e Dependência Estratégica.

Finalizando, a última subseção apresenta o CLIPS - C Language Integrated Production System.

2.1 Léxico Ampliado da Linguagem (LAL)

A geração do léxico tem por objetivo geração de conhecimento em nível de produto, e o foco é o que se chama de Léxico Ampliado da Linguagem (Leite e Franco, 1993). O LAL é um metamodelo utilizado para as fases de elicitação dos requisitos. A ideia do modelo é centrada na descrição de termos da linguagem que auxiliam o entendimento do domínio do problema, de onde o software será derivado. O LAL se baseia na premissa de que em uma organização existe uma linguagem, utilizada pelos atores da organização e é estendida da linguagem natural. Essa linguagem, com vocabulário peculiar (composto de símbolos), é denominada linguagem da aplicação (Leite e Franco, 1993). O Léxico tem como justificativa uma razão simples: "entender a linguagem do domínio sem se preocupar com o entendimento do problema envolvido". O objetivo do LAL é capturar o vocabulário (termos e sentenças) intrínseco ao contexto organizacional.

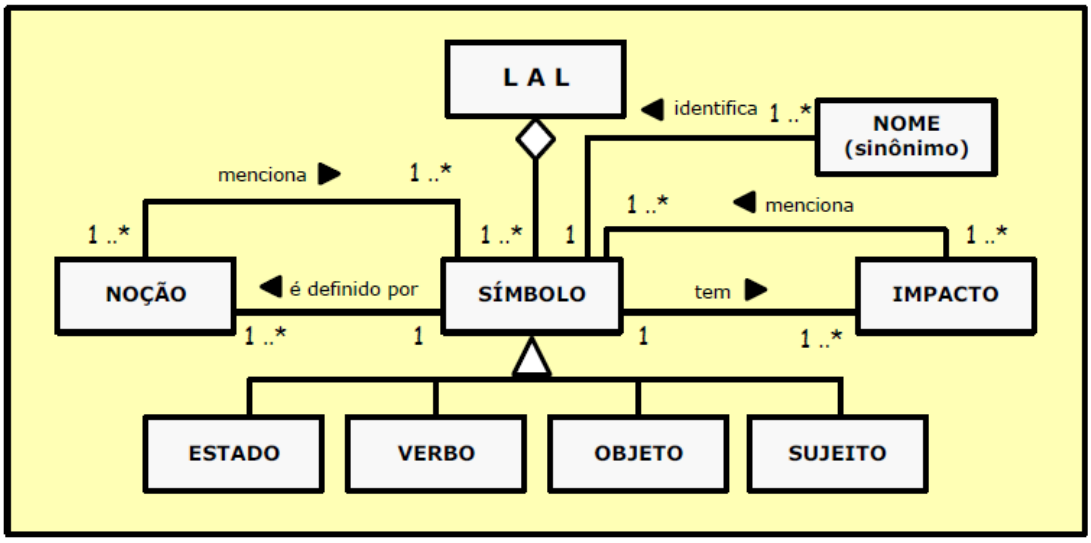


Figura 1. Diagrama de classe dos elementos do LAL

Fonte: (Oliveira, 2008).

O LAL é formado por símbolos e cada um deles é identificado por um nome. A Figura 1 apresenta os componentes do LAL. Um símbolo pode possuir um nome, sinônimos e duas descrições. A descrição chamada noção, que descreve o significado real do símbolo em questão, representa o sentido denotativo do termo (Oliveira et al.,

2006). A segunda descrição, chamada de impactos ou resposta comportamental, descreve o sentido conotativo do termo, isto é, descreve como o símbolo age dentro do contexto organizacional e seus efeitos comportamentais no contexto. Ambos, noção e impacto, fazem menção a outros símbolos. Além disso, os símbolos do LAL podem ser classificados em quatro tipos: sujeito, verbo, objeto e estado.

A construção do LAL é baseada em dois princípios básicos: circularidade e vocabulário mínimo. A circularidade é a ideia de descrever termos em função de outros já definidos. É recomendado maximizar o uso de termos do léxico nas descrições de outros. O vocabulário mínimo é a ideia de minimizar o uso de palavras fora do léxico, isto é, quanto menos termos fora do contexto da aplicação forem usados nas descrições, melhor (Leite e Franco, 1993).

2.2 Framework de modelagem i*

A técnica do i* foi proposta por Eric Yu em sua tese de doutorado (Yu, 1995) com o objetivo de fornecer uma técnica que propiciasse a representação conceitual para descrever processos que envolvam diversos participantes. Apesar da existência de técnicas capazes de fornecer tais modelos, a maioria dessas técnicas de modelagem tem como principal foco aspectos comportamentais do processo, deixando de lado aspectos motivacionais relacionados ao comportamento. Em geral, as técnicas permitem a descrição dos componentes do sistema (atores), capacidades (processos que podem executar) e comportamentos (como e quando os processos são executados), mas não conseguem expressar as razões envolvidas nos processos (o porquê). Já o i* fornece conceitos para modelar e responder questões como:

- Por que atores executam os processos?
- Quem deseja que eles façam isso?
- Quais são as formas alternativas de executar um processo?
- Por que os atores possuem ou recebem informação?

Uma análise e entendimento mais profundos sobre as motivações, os "porquês" ou razões são passos fundamentais no processo de entendimento do domínio. Apesar da importância, esse conhecimento não pode ser expresso na maioria dos modelos

convencionais, que apenas permitem a descrição de entidades e atividades de uma organização e os seus relacionamentos.

A modelagem intencional com i* modela contextos organizacionais com base nos relacionamentos de dependência estratégica entre os atores desse contexto, permitindo a representação em diagramas i* das metas que os atores precisam atingir. Para o i*, "A meta é uma condição ou estado de coisas do mundo que um ator gostaria de alcançar" (Yu, 1995). Na representação do i*, atores dependem uns dos outros para que metas sejam alcançadas, recursos sejam fornecidos, tarefas sejam realizadas e metas flexíveis (softgoals) sejam "razoavelmente satisfeitas". O i* utiliza o mesmo conceito usado por (Chung et al., 2000) de que uma meta flexível (requisito não-funcional - RNF) possui uma característica de incerteza para a sua satisfação ou não.

Para o i*, tarefas (tasks) são os principais meios (means) para que metas (ends) sejam atingidas. Por outro lado, pode-se identificar que existem dois tipos de tarefas ou ações: a primeira como definido por Sá Carvalho (Carvalho, 1988) é uma ação concreta, que é executada em um contexto organizacional e pode ser descrita por um verbo ativo, concreto e bem definido (exemplos: comprar, vender, cobrar, multar, produzir, treinar, alocar, etc.). Para Carvalho (Carvalho, 1988), uma ação concreta não pode ser descrita por um verbo pouco preciso (como controlar, apurar, administrar, acompanhar, etc.). No segundo tipo, as tarefas ou ações definidas por Oliveira (Oliveira, 2008), por exclusão da definição anterior, podem ter a qualificação de flexível, sendo que uma ação flexível tem o mesmo significado ou interpretação usados para metas flexíveis (softgoals). As ações flexíveis são pouco precisas, para as quais não se pode identificar um resultado concreto a priori e, além disso, a confirmação da execução da ação flexível pode depender de interpretação. Uma ação flexível (exemplos de "ações flexíveis": analisar, apurar, avaliar, conferir, controlar, gerenciar, verificar, validar, etc.) é diferente de uma ação concreta (Oliveira, 2008).

O i* é composto de dois tipos de modelos: o Modelo de Dependências Estratégicas (SD) e o Modelo de Razões Estratégicas (SR). O Modelo SD fornece uma descrição intencional de um processo em termos de uma rede de relacionamentos de dependência entre atores. O Modelo SR fornece uma descrição intencional do processo em termos de seus elementos e as razões que estão por detrás deles, seja por meio da modelagem de seus atores, das metas (concretas e flexíveis) que precisam atingir e de como pretendem alcançá-las (recursos e tarefas).

2.2.1 Modelo SD

O modelo de Dependência Estratégica (SD) consiste em um conjunto de ligações, em que cada nó representa um ator e a ligação entre dois atores indica uma relação de dependência entre eles. A dependência pode ser não somente física, mas também informacional. Nessa relação, temos que um ator que é dependente de um segundo ator é denominado depender. O ator de quem se depende é chamado de dependee. Mas para que haja a dependência, deve haver um objeto de dependência que recebe o nome de dependum. O dependum pode ser de quatro tipos: meta concreta, meta flexível, tarefa ou um recurso. Devido à relação entre os atores, o depender torna-se vulnerável, visto que pode ocorrer o não fornecimento do objeto de dependência por parte do dependee.

Se o objeto de dependência for uma meta concreta, o depender depende do dependee para a realização de uma determinada meta. As metas são estados ou condições nas quais o ator gostaria de chegar ou alcançar. As metas concretas são condições claramente definidas de forma que se possa avaliar a sua satisfação de forma clara e objetiva. Neste caso, cabe apenas ao dependee decidir questões relativas para que a meta seja cumprida. A mostra Figura 2 essa relação.

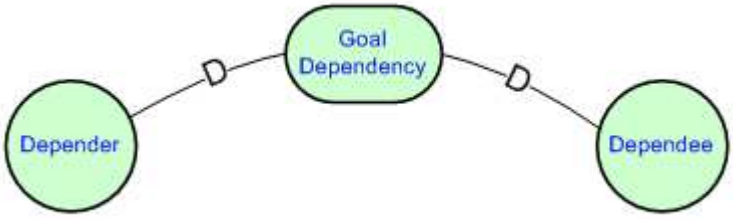


Figura 2. Dependência por meta.

Fonte: (Wiki, 2013).

Já a Figura 3 representa um relacionamento de dependência em que o dependum é uma tarefa. Uma tarefa especifica a realização de alguma atividade. Na dependência de tarefa, o dependee é solicitado a executar uma determinada atividade. O depender toma as decisões de como a tarefa deve ser executada, apesar da descrição da tarefa não necessariamente conter todos os passos para que seja executada.

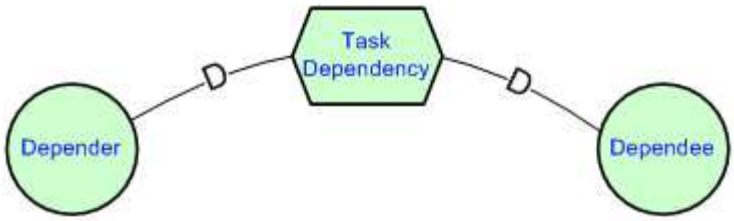


Figura 3. Dependência por tarefa.

Fonte: (Wiki, 2013).

Em outro cenário, o dependum pode ser caracterizado como um recurso. Um recurso é caracterizado não só por uma entidade física, mas também pode ser uma informação, sendo o produto final de algum processo ou atividade que poderá estar disponível ao depender. Nesse caso, não existe a necessidade de decisões serem tomadas a respeito da produção do recurso, que deve ser fornecido ao depender exatamente da maneira ordenada ao dependee. A Figura 4 ilustra esta representação.

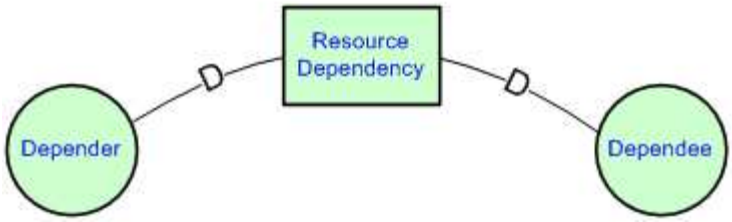


Figura 4. Dependência por recurso.

Fonte: (Wiki, 2013)

Por último, apresentada pela Figura 5, existe a dependência por meta flexível (softgoal). A meta flexível é um estado ou condição que o ator deseja alcançar, diferindo da meta concreta em relação ao critério de aceitação. Tais metas não possuem critérios claramente definidos. Com isso, na dependência por meta flexível, a aceitação do cumprimento da meta pelo depender é subjetiva e depende apenas do seu próprio julgamento.

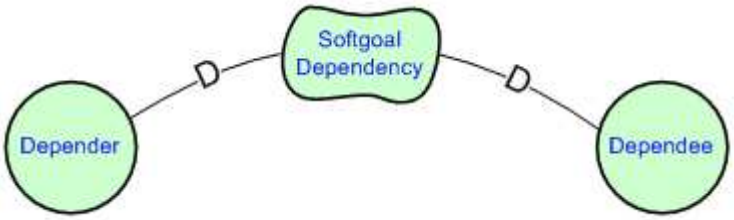


Figura 5. Dependência por meta-flexível.

Fonte: (Wiki, 2013).

Todas as dependências do modelo SD ainda possuem graus de dependência ou vulnerabilidade diferentes, apresentados graficamente pela Figura 6 (Grau et al., 2008):

- Comprometida ("committed"): graficamente não possui sinais. É usada quando falhando o relacionamento entre os atores, as intenções são afetadas, mas não prejudicadas.
- Aberta ("open"): graficamente sinalizada com "O". É usada quando havendo falha na relação de dependência, as consequências não são relevantes.
- Crítica ("critical"): representada graficamente pelo símbolo "X". Sua utilização se dá quando na falha da relação de dependência, todos os relacionamentos e dependências são abalados, pois afetam de forma grave as intenções.

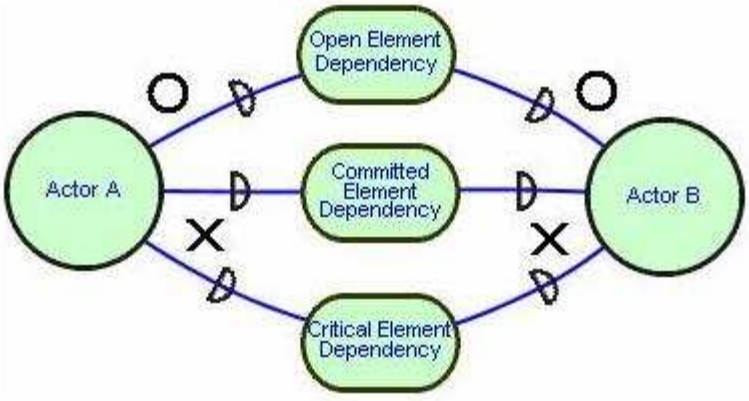


Figura 6. Representação dos graus de dependência.

Fonte: (Wiki, 2013).

2.2.2 Modelo SR

O modelo de Razão Estratégica (SR) descreve o relacionamento dos atores entre si, além da razão ou motivações que cada ator tem sobre suas metas em termos dos elementos do processo. O modelo SR descreve as razões associadas ao processo pelo qual as metas são alcançadas, tarefas são elaboradas, recursos são entregues e metas flexíveis são decompostas e atingidas. O modelo SR permite um refinamento da descrição dos processos em relação aos modelos SD. Assim, as tarefas executadas são detalhadas, onde podem ser encontradas maneiras alternativas para a concretização dos objetivos.

O modelo SR é composto pelos mesmos elementos e ligações do modelo SD, contudo são incorporadas representações dos relacionamentos intencionais internos aos atores, os relacionamentos meio-fim e as ligações de decomposição de tarefas.

As ligações de decomposição de tarefas são ligações em que um nó tarefa é ligado aos nós componentes, que podem ser outras tarefas, metas, recursos ou metas flexíveis. Graficamente, tais ligações são representadas como mostrado na Figura 7. Uma tarefa decomposta só pode ser considerada completa assim que todos os componentes forem realizados ou satisfeitos. São considerados quatro tipos de classes de ligações:

- Objetivo – Tarefa: quando uma meta é um componente da tarefa, implica que a meta representa uma etapa necessária na ação representada pela atividade realizada na tarefa.

- Tarefa – Tarefa: quando uma tarefa é atribuída como subtarefa de outra, a subtarefa representa um fluxo necessário na composição da tarefa superior.

- Recurso – Tarefa: no caso de uma tarefa ter como componente um recurso, isso significa que a tarefa demanda tal recurso na sua realização.

- Meta flexível – Tarefa: neste tipo de ligação onde uma meta flexível é um componente de uma tarefa, significa que a meta flexível serve como um objetivo de qualidade para a tarefa.

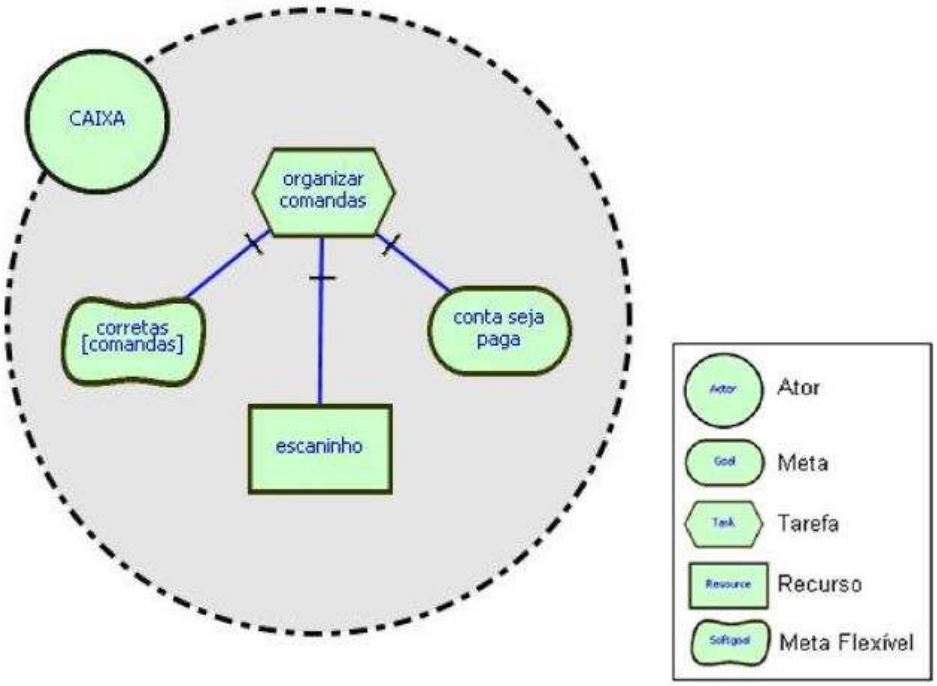


Figura 7. Elo de decomposição de tarefa.

Fonte: (Wiki, 2013).

Já a relação meio-fim é graficamente apresentada na Figura 8. O nó fim que é apontado pelo vetor, geralmente trata-se de uma meta a ser alcançada, podendo ser uma meta flexível, um recurso ou até mesmo uma tarefa. Já o nó meio, normalmente trata-se de uma tarefa ou mais, nos casos de tarefas alternativas e exclusivas, visto que em uma tarefa se espera que algo seja feito.

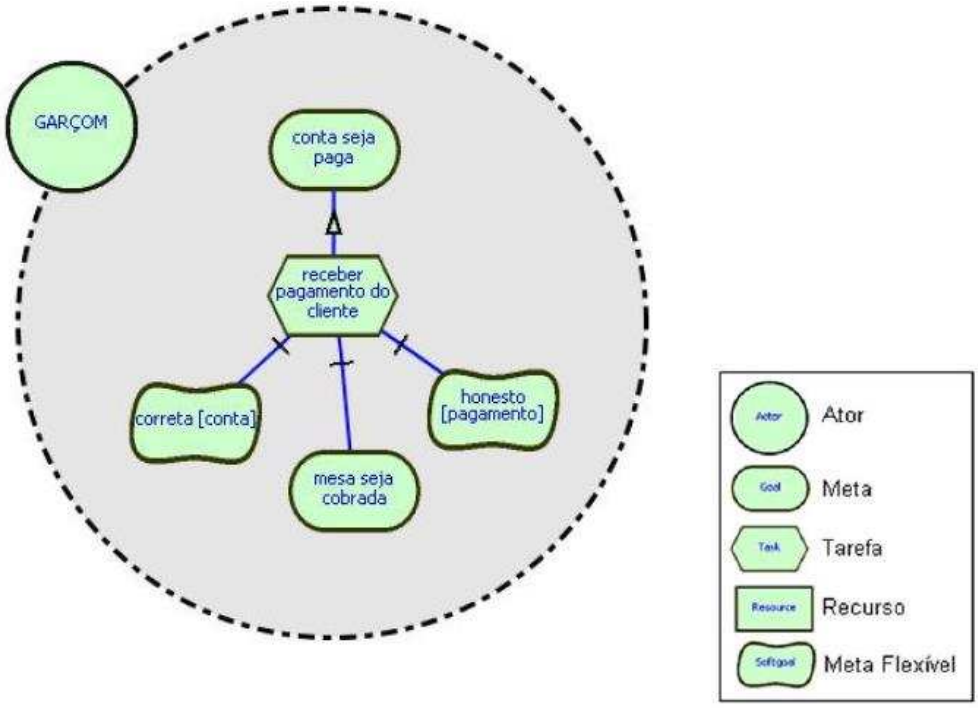


Figura 8. Elo meio-fim.

Fonte: (Wiki, 2013).

No modelo SR existem também os links de contribuição. São links usados para representar a contribuição de um determinado elemento em relação a uma meta flexível. Esses links representam a forma como os elementos contribuem com uma determinada meta flexível, podendo ser de forma positiva, negativa ou neutra. Existe ainda a escala de contribuição: make (“++”), help (“+”), unknown (“?”), hurt (“–”) e break (“– –”).

2.2.3 Modelo AS

O modelo de Atores Estratégicos, ou Modelo AS, modela os atores do contexto, facilitando o entendimento dos relacionamentos entre eles. Segundo Yu (Yu, 1995), o ator é uma entidade ativa que executa atividades para atingir metas por meio de seu know-how. O termo “ator” é utilizado para referenciar genericamente qualquer unidade a que dependências intencionais possam ser atribuídas. Atores podem ser considerados: intencionais, por possuírem motivações na realização de suas ações, e estratégicos, por não focarem exclusivamente nas suas metas imediatas, mas se preocupam com as implicações de seu relacionamento com outros atores.

Dentro do modelo SA existem alguns conceitos importantes que são: agente, posição e papel. A seguir é definido cada um dos conceitos (Wiki, 2013):

I. O agente é um ator que possui manifestações concretas e físicas. O agente pode referir-se a humanos ou a entidades computacionais, como software ou hardware. O agente ocupa posições e desempenha papéis, e papéis são cobertos por posições.

II. O papel é uma caracterização abstrata do comportamento do ator no contexto social. Apresenta as funções que podem ser exercidas por um agente da organização.

III. A posição é uma entidade intermediária entre um agente e um papel, sendo o conjunto de papéis ocupados por um agente.

Existem alguns links que servem para representar as relações entre atores. A Tabela 1 define os tipos de possíveis associações (Wiki, 2013). A Figura 9 mostra graficamente os elementos do modelo AS. Todos esses conceitos são apresentados em um estudo detalhado realizado por Leite et al. (Leite et al., 2007). A Figura 9 também apresenta a notação gráfica para os elos de relação dos elementos no modelo AS.

Tabela 1. Elos dos elementos no modelo SA.

Is part of	É o relacionamento de agente para agente, posição para posição e de papel para papel, indicando parte e o todo.
Is a	É o relacionamento usado para indicar especialização.
Plays	É o relacionamento de agente para papel, indicando que um agente desempenha um papel.
Occupies	É o relacionamento de agente para posição, indicando que um agente ocupa uma posição.
Ins	É o relacionamento de agente real para agente, indicando que um agente é uma instância de outro.
Covers	É o relacionamento de posição para papel, indicando quais papéis são cobertos por uma posição.

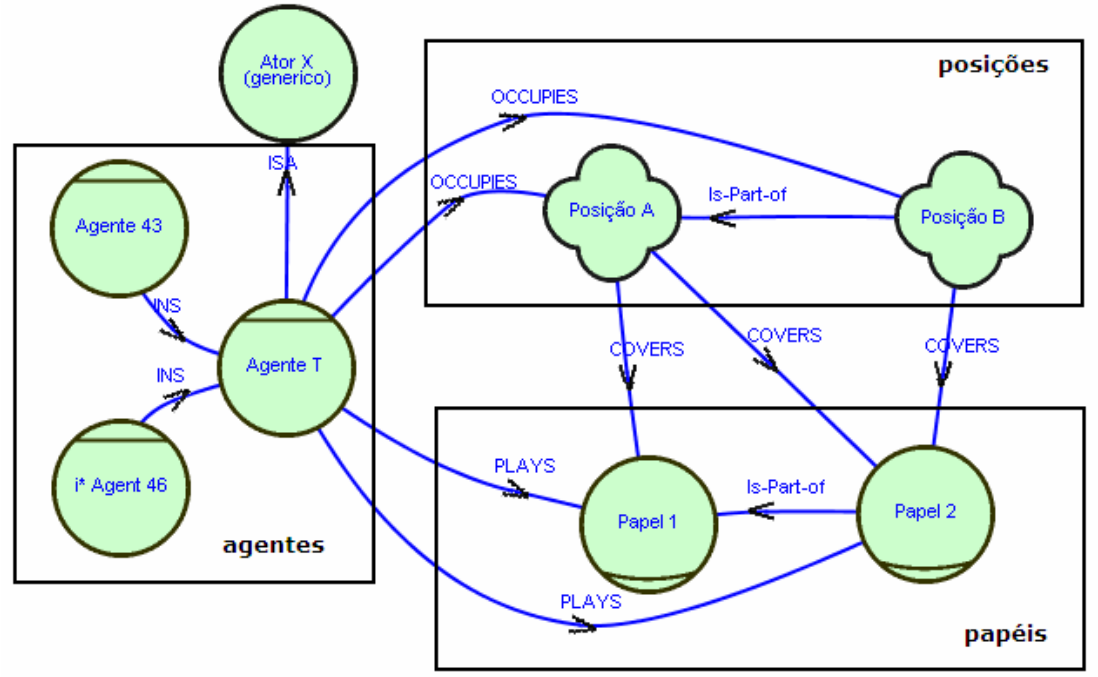


Figura 9. Exemplo de Modelo AS.

Fonte: (Oliveira, 2008).

2.3 Sistemas Especialistas

O conhecimento pode ser prático ou teórico a respeito de um assunto ou domínio. Quem possui um grande conhecimento a respeito de um determinado domínio é denominado especialista. Os sistemas especialistas são simuladores do comportamento de uma especialista em determinadas situações (Negnevitsky, 2005).

Os sistemas especialistas são baseados na representação do conhecimento humano. Apesar da dificuldade de se representar a forma como ocorre o processo mental de um humano, os sistemas especialistas assumiram que o ser humano raciocina por meio de regras de produção para resolução de problemas. Todo o conhecimento armazenado na memória humana, submetido a regras de produção, seria capaz de chegar a conclusões a respeito de um problema.

As regras possuem um formato definido, uma relação entre o antecedente e o consequente. O antecedente é composto pelas condições necessárias para que uma determinada regra seja disparada. Já o consequente é composto por ações que são realizadas dado que as premissas do antecedente foram satisfeitas.

O antecedente possui o formato **se** X **então** Y. Para exemplificar, considere a seguinte regra: **se** o tanque está vazio **então** o carro está parado. A parte antecedente pode combinar os conectivos lógicos e, ou e não para expressar a relação entre as premissas. Na parte consequente, são encadeadas as ações realizadas quando a regra for disparada e as premissas são satisfeitas. Assim, com a junção de conectores é possível produzir regras que levam em consideração mais de uma premissa e possua mais de uma ação compondo seu consequente. O formato genérico é o seguinte:

SE
antecedente$_1$ **OU|E|NÃO** antecedente$_2$, ..., **OU|E|NÂO** antecedente$_n$
ENTÃO
consequente$_1$, consequente$_2$, ..., consequente$_n$

Existem algumas características fundamentais em sistemas especialistas como (Negnevitsky, 2005):

- A precisão da resposta. Os sistemas simulam o conhecimento humano muito profundamente, então não importa a rapidez com que o sistema solucione o problema, a menos que o solucione corretamente.
- O sistema especialista deve ser explicativo, ou seja, capaz de raciocinar e explicar suas decisões. A explicação seria a linha de raciocínio utilizada, ou seja, as regras disparadas para atingir determinada conclusão.
- Diferentemente dos sistemas convencionais, sistemas especialistas usam raciocínio simbólico ao invés de processamento numérico. Além disso, sistemas convencionais sempre executam operações passo a passo bem definidas. Diferentemente, sistemas especialistas permitem o raciocínio inexato e podem lidar com dados incompletos e imprecisos.

Todas as características são incorporadas nos componentes do sistema especialista. A Figura 10 mostra a estrutura de um sistema especialista. A representação e inferência do conhecimento pode ser dividida em 5 componentes:

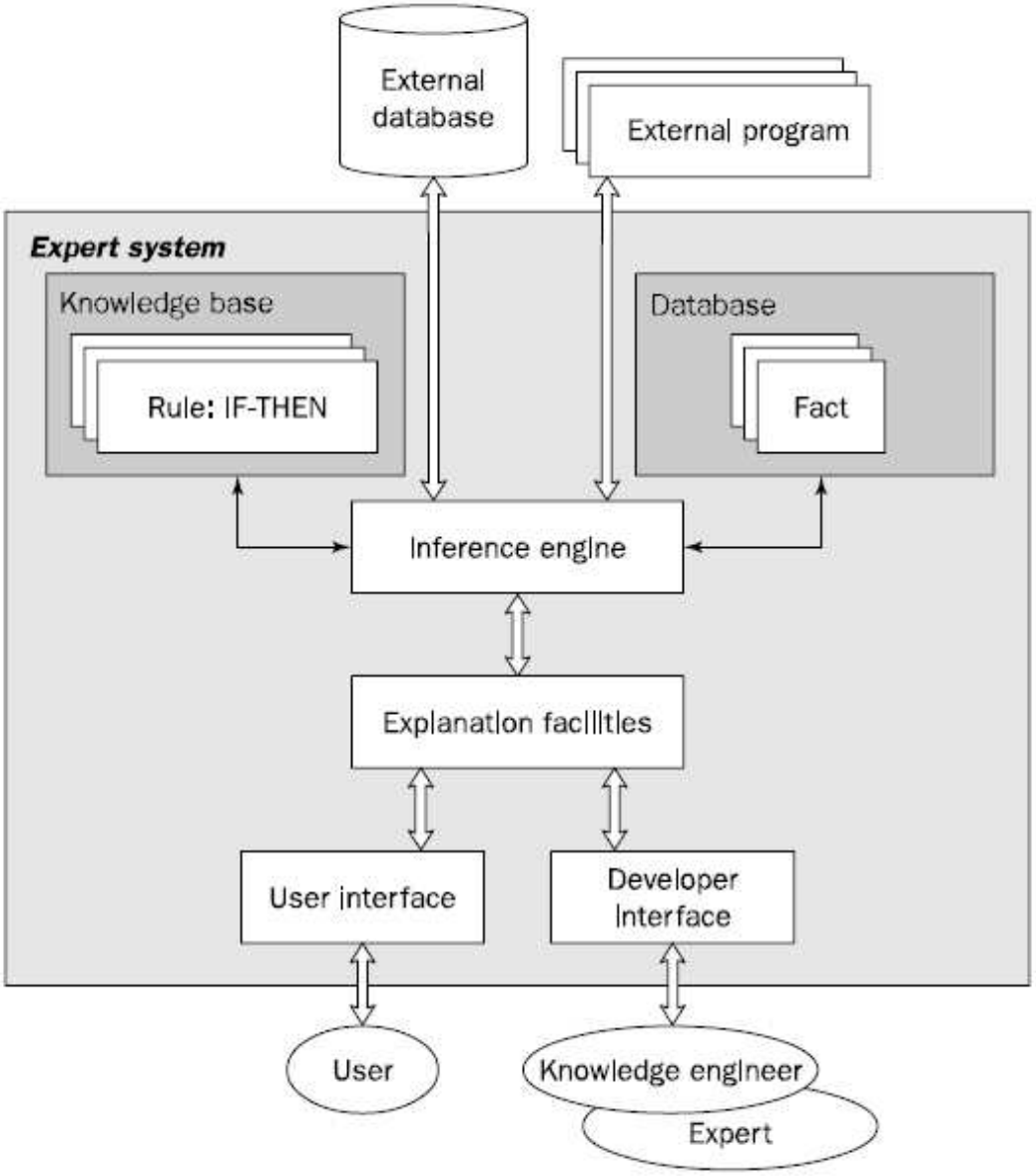


Figura 10. Estrutura de sistemas especialistas baseado em regras.

Fonte: (Negnevitsky, 2005).

- **Knowledge Base**: Todo o conhecimento útil para a resolução do problema se encontra na base de conhecimento. Todo o conhecimento é representado por meio de um conjunto de regras. As regras especificam relações, recomendações, diretivas, estratégias ou heurísticas no formato **SE** (condição) **ENTÃO** (ação).

- **Database**: Inclui o conjunto de fatos usados para a investigação por parte das regras, os dados a respeito do problema.

- **Inference Engine**: O motor de inferência é o responsável por raciocinar sobre o conhecimento, atingindo uma solução além ligar os fatos com base de conhecimento.

- **Explanation facilities**: Possibilita o usuário a concluir como uma solução particular foi alcançada e porque um fato específico é necessário.

Os cinco componentes são essenciais a um sistema especialista, mas existem componentes adicionais que os tornam mais completos. Em algumas situações são necessárias interações com outros sistemas, como no caso de processamento externo ou no caso de extração de dados armazenados em bases externas. Existem também as necessidades de interação com os programadores. Assim, normalmente os sistemas especialistas contam também com uma interface voltada ao desenvolvedor para edição da base de conhecimento ou mesmo da base de fatos, além de ferramentas de listagem de regras disparadas. Interações com os usuários também são uma preocupação dos sistemas especialistas, visto que é possível a inserção de conhecimento em tempo de execução.

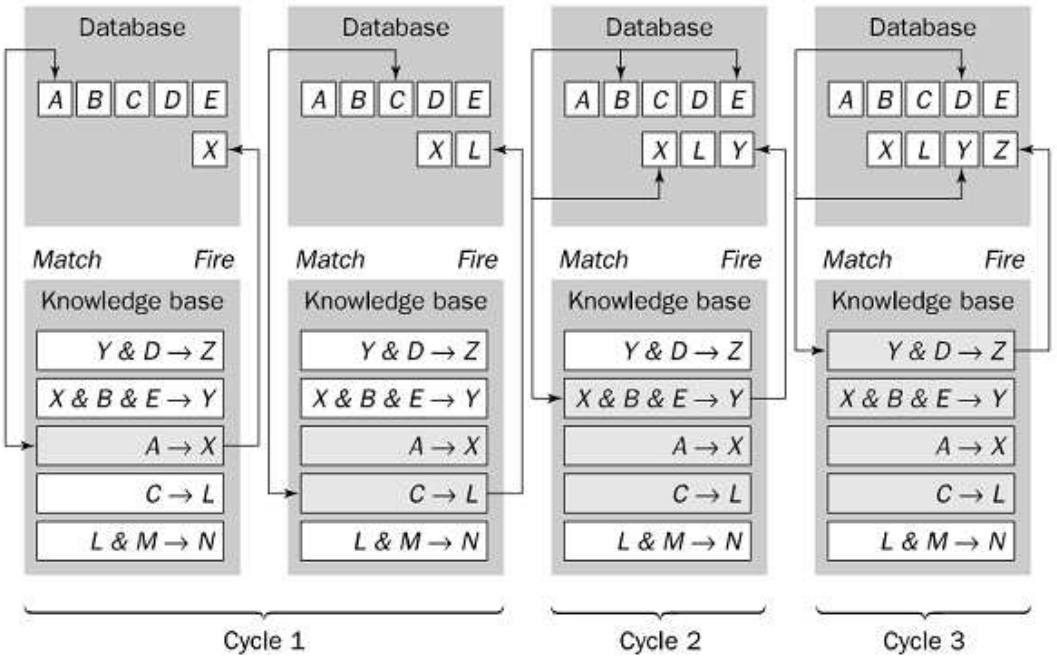
O mecanismo de inferência de um sistema especialista, representado pelo componente inference engine, busca regras de produção que são aplicadas aos fatos a fim de se chegar a uma conclusão. São disparadas ações nas quais os antecedentes são satisfeitos pelos dados da base, o que pode levar à adição de conhecimento à base. Durante o processo de investigação são utilizadas duas estratégias básicas: forward chaining ou backward chaining.

A estratégia forward chainning ou "encadeamento para frente" é uma técnica que agrupa o conhecimento e infere a partir do grupo de informações obtidas. Nesse caso, é possível a adição de conhecimento a cada regra disparada, o que exige uma reavaliação de todas as regras, podendo gerar o disparo de regras que não fazem parte da linha de raciocínio para conclusão desejada.

A Figura 11 exemplifica o fluxo deste tipo de encadeamento. Inicialmente, duas regras possuem suas premissas atendidas, finalizando o ciclo e adicionando aos fatos X e L. Com isso, mais uma regra tem suas premissas satisfeitas, o que dispara outra regra e adiciona o fato Y à base. Esse processo é feito ciclicamente até que se chegue a uma conclusão desejada. Esse encadeamento é recomendado para análise e

interpretação. Em casos em que a conclusão só pode ser obtida a partir de um fato particular, é sugerida a técnica de backward chaining, recomendada para sistemas de diagnósticos (Negnevitsky, 2005).

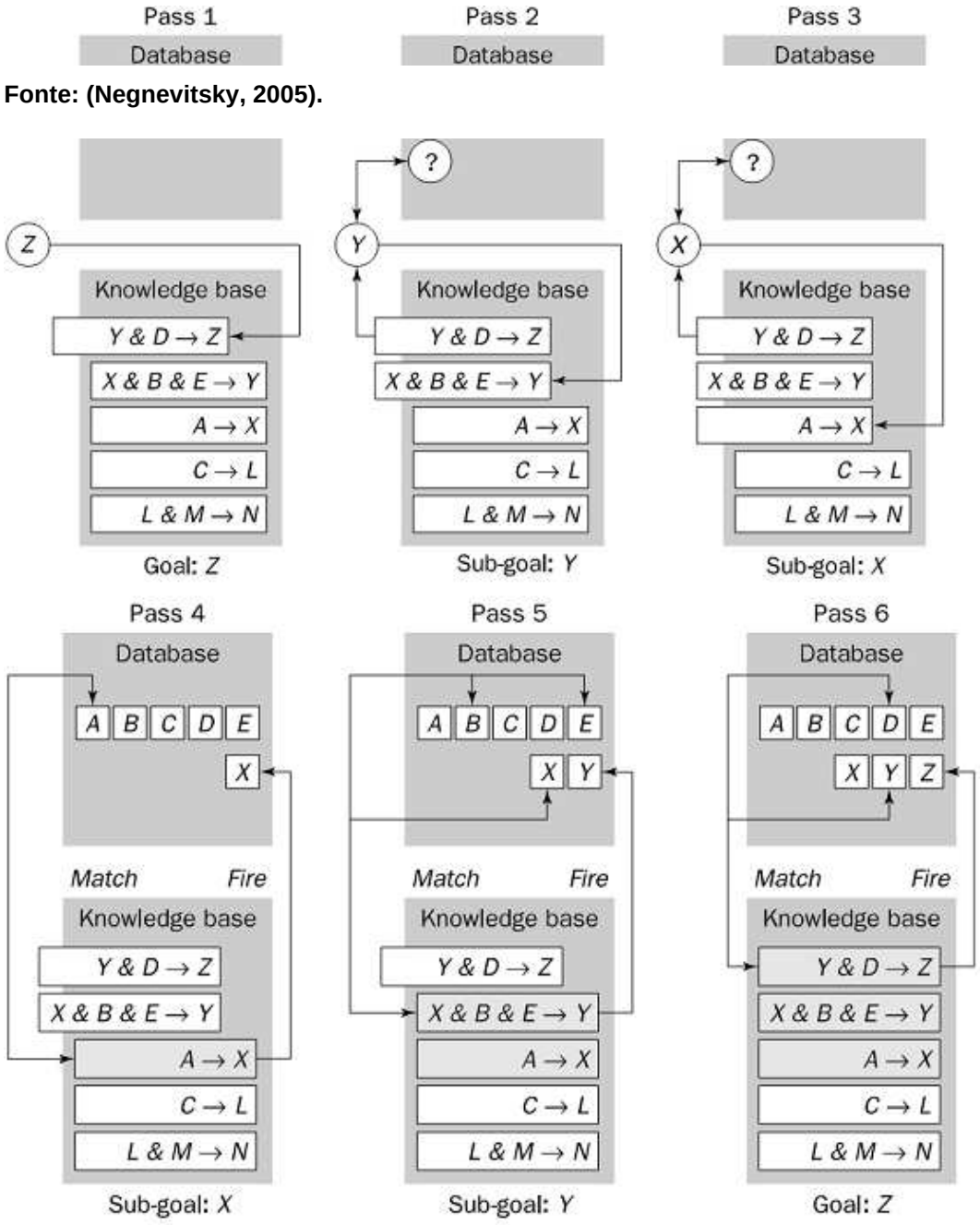
Figura 11. Técnica do encadeamento para frente.



A Figura 12 exemplifica a citada backward chaining ou "encadeamento para

Fonte: (Negnevitsky, 2005).

trás". A partir de um objetivo, o sistema busca evidências para provar a solução hipotética. São buscadas regras na base que possuem como consequente o objetivo esperado. Feito isso, são empilhadas tais regras e o objetivo passar a ser as premissas das regras empilhadas e assim por diante, até que não se tenha premissas a serem atingidas.



Fonte: (Negnevitsky, 2005).

Figura 12. Técnica do encadeamento para trás.

Na Figura 12 temos que Z é a solução do problema, então é examinada a base em busca de regras que possuem em seus consequentes Z. A primeira regra da Figura 12 tem o fato Z como ação, porém Y e D não fazem parte da base de fatos. Assim, a regra Y & D→Z é empilhada. O próximo passo é tentar provar Y, depois provar D e, finalmente, provar Z. A segunda regra prova Y, porém X não está na base, então se empilha essa regra e prova-se X pela análise da quarta regra. Uma vez provado X, a

segunda regra é provada e desempilhada e, finalmente, a primeira regra é disparada, provando assim que Z é solução para o problema.

2.4 CLIPS

O ambiente de desenvolvimento escolhido para aplicar as técnicas de Inteligência Artificial é o CLIPS - C Language Integrated Production System (Giarratano, 1993). O CLIPS é uma ferramenta para sistemas especialistas, desenvolvida pelo Software Technology Branch (STB), NASA/Lyndon B. Jhonson Space Center. Existem três maneiras para representar o conhecimento por meio do CLIPS:

1. Regras, conhecimento heurístico baseado na experiência;
2. Funções que são destinados ao conhecimento procedimental;
3. Programação Orientada a Objetos, voltada também para o conhecimento procedimental.

CLIPS foi desenvolvido com o propósito de facilitar o desenvolvimento de sistemas especialistas. O conhecimento é representado por meio de regras de produção e a base de dados é composta de fatos. O mecanismo de inferência é baseado na técnica de encadeamento progressivo (forward chaining) apresentado anteriormente.

A escolha do uso de CLIPS é devido a algumas vantagens e características mostradas por Negnevitsky (Negnevitsky, 2005), visto que a linguagem é usada para construção de sistemas especialistas baseados em conhecimento. Como CLIPS é uma linguagem que pode ser considerada simples, não é necessária uma habilidade adicional em programação. Outra característica presente no CLIPS é a facilidade com que são tratadas entradas e saídas em tempo de execução. Tal característica permite ao sistema adicionar novas informações durante a execução a sua base de dados.

Essas características são igualmente interessantes na abordagem do problema citado neste trabalho. A opção de uma programação de baixa complexidade busca focar em aspectos que interessam ao domínio do estudo (Oliveira et al., 2013).

2.5 Método ERi*C

Nesta seção, é apresentado o método para o qual a ferramenta proposta nesse trabalho fornece suporte. Todo o fluxo de elaboração e construção da ferramenta é baseado no processo de construção de modelos mostrados pelo método Engenharia de Requisitos Intencional (ERi*c) (Oliveira, 2008). O método ERi*c é composto por 6 etapas distintas. A Figura 13 apresenta resumidamente quais são as etapas para elaboração dos modelos.

Na primeira etapa, o método ERi*c descreve uma técnica para elicitação de metas concretas e metas flexíveis dos atores organizacionais. Assim, é usada linguagem do domínio representada por meio das descrições do LAL (Leite e Franco, 1993). A técnica denominada "Metas dos Agentes vindas do Léxico" foi proposta também por Oliveira em (Oliveira et al., 2007).

Já na etapa seguinte, os esforços são para identificação das situações de dependência estratégica (SDsituations) (Oliveira e Cysneiros, 2011) e, em seguida, a modelagem das metas dos agentes. Nesta etapa, é observado como as metas compõem conjuntos na formação de situações de negócio.

Na terceira etapa, são apresentados o conceito de Painel de Intencionalidade ou "Intentionality Panel" – IP Diagram (Oliveira et al., 2007). Nesta etapa, são agrupadas as metas e essas são relacionadas umas com as outras, sendo construído um diagrama com as contribuições, relações de dependência e demais relacionamentos entre as metas.

Na quarta etapa, são construídos modelos simplificados dos modelos originais. Nesses novos modelos, chamados de SRcronstucts (Oliveira et al., 2007), são detalhadas as metas concretas e flexíveis. Outra contribuição dada nesta etapa é a orientação para derivação de modelos SR a partir de modelos SD.

Na penúltima e última etapa são apresentadas heurísticas para a especificação das SDsituations e são realizadas verificações nos modelos SD e SR, respectivamente.

O objetivo desta dissertação é apresentar uma proposta de ferramenta que contemple e dê suporte ao ER que opte pelo método ERi*c, para a construção de modelos orientados a agentes. Mais especificamente, o trabalho se limita e tem como foco a primeira etapa do método. No decorrer da dissertação, é apresentado o passo a passo da utilização da ferramenta. São apresentados exemplos e produtos ao final de cada tarefa realizada dentro da primeira etapa do método.

Originalmente, a primeira etapa do método possui apenas três tarefas a serem realizadas:

1. Preparar o LAL
2. Definir AGFL – Metas dos Agentes vindas do Léxico
3. Refinar as metas

A Figura 14 apresenta o fluxo das atividades originais presentes na primeira etapa.

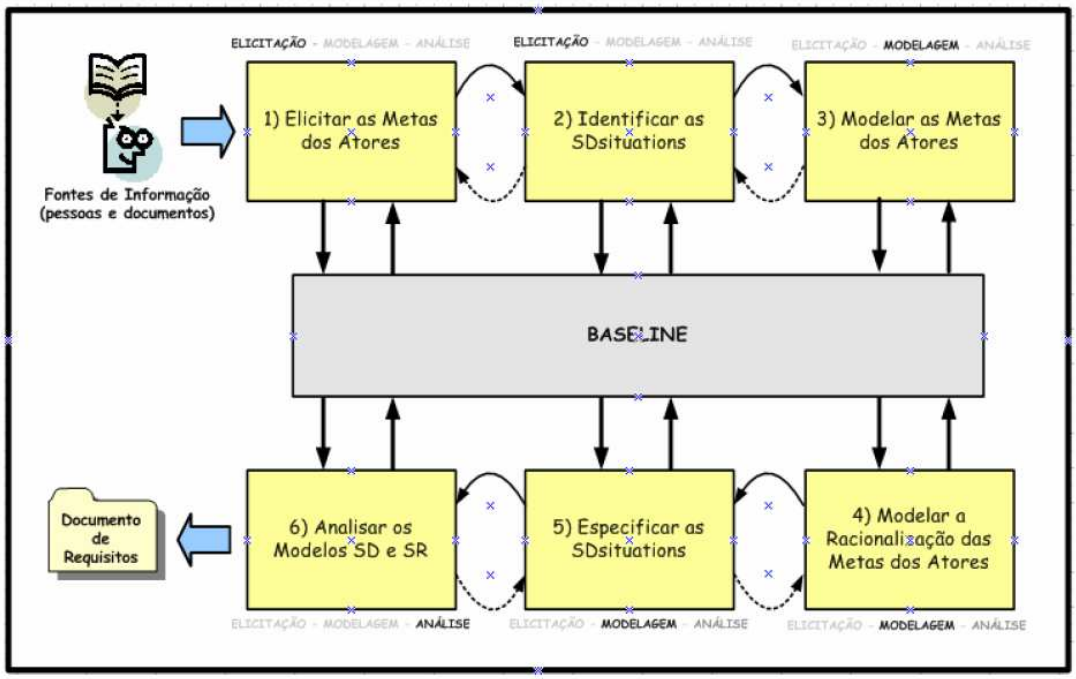


Figura 13. Etapas do método ERi*c.

Fonte: (Oliveira, 2008).

Para a utilização da ferramenta como apoio ao método, foram introduzidas três novas tarefas intermediárias às originais:

- Estruturar Conhecimento do LAL
- Capturar Aprendizado

- Exportar as Metas Elicitadas

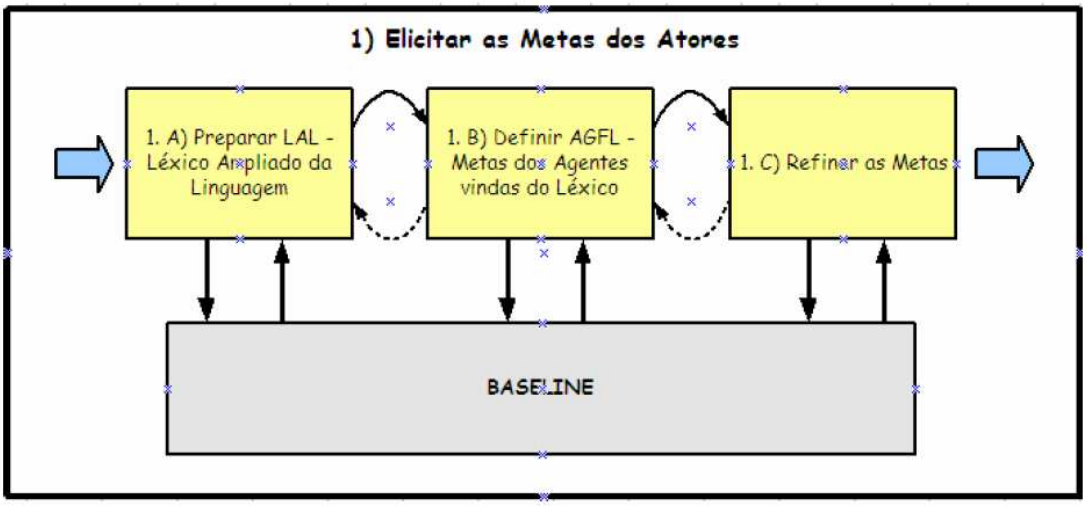


Figura 14. Tarefas da primeira etapa do método ERi*c.

Fonte: (Oliveira, 2008).

A Figura 15 mostra a nova configuração do método com o uso da ferramenta. As atividades originais do método agora são totalmente ou parcialmente automatizadas. A Figura 15 apresenta, em verde, atividades presentes no método original. Essas atividades foram incorporadas à ferramenta. Em azul, são apresentadas as novas atividades inseridas no método, necessárias para a automação da extração das metas. O detalhamento, os produtos e os passos provenientes das tarefas a serem executadas são apresentados na seção seguinte.

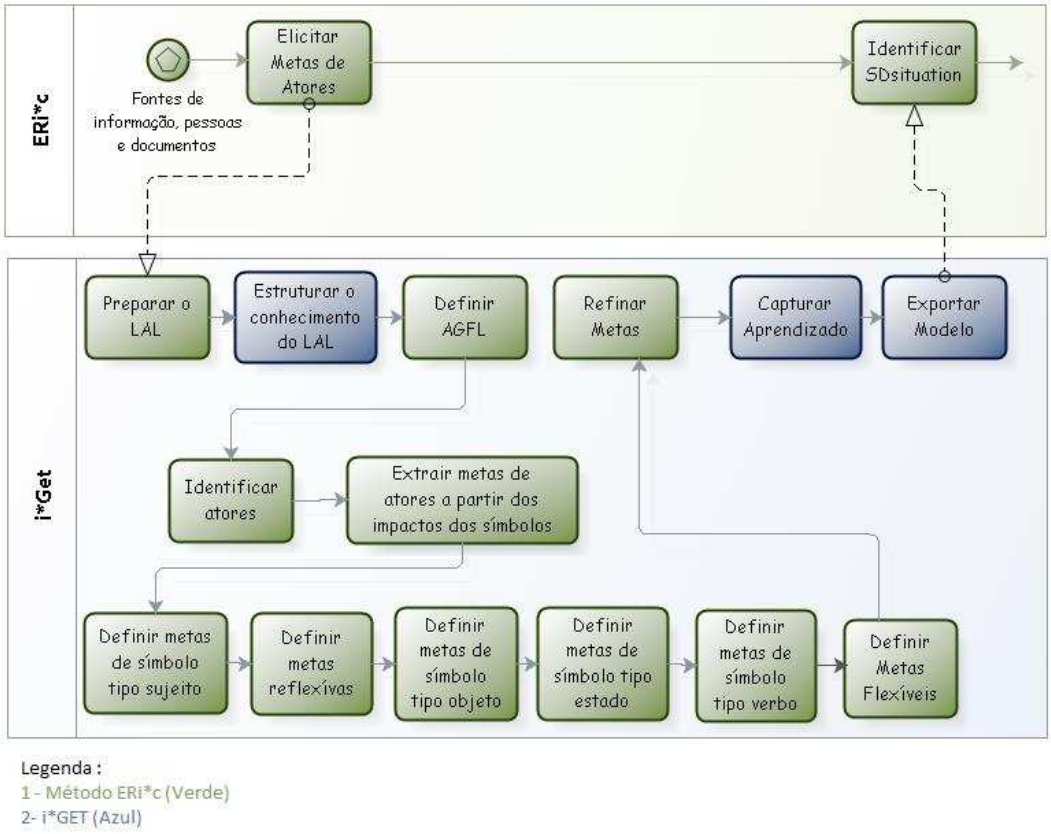


Figura 15. Fluxo completo de atividade inseridas no método ERi*c.

Fonte: Elaborada pelo autor.

3 A FERRAMENTA i*GET

A ferramenta batizada de i* Goal Elicitation Tool – i*GET leva em consideração alguns passos e atividades a serem realizadas dentro do método. Para o melhor entendimento, será usado um exemplo com a utilização da i*GET, que será apresentado na seção EXEMPLO DE APLICAÇÃO.

A Figura 16 mostra o diagrama SADT (Structured Analysis and Design Technique) (Marca e Mcgowan, 1987) da i*GET. O diagrama apresenta uma visão geral sobre todo o processo. As informações contidas nas descrições do LAL são submetidas à i*GET. Sob o controle das regras de produção e sintaxe do CLIPS, as informações passam pela i*GET, produzindo ao fim ações que alimentam a base de conhecimento, além de fornecer as metas elicitadas no formato iStarML (Cares et al., 2011).

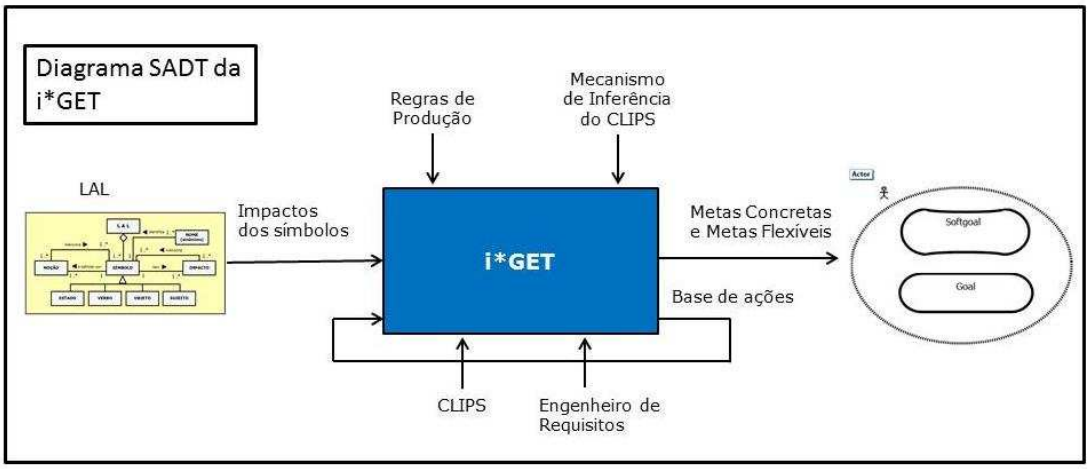


Figura 16. Diagrama SADT do modelo conceitual da ferramenta.

Fonte: Elaborado pelo autor.

O método proposto por Oliveira (Oliveira et al., 2008a) possui algumas etapas distintas. Dessa forma, foram criados módulos dentro da i*GET que são responsáveis por desempenhar atividades relativas a cada uma das etapas apresentadas na seção anterior. Os módulos foram criados para facilitar a usabilidade do sistema, já que existe uma forte interação com o ER. O ER tem liberdade de navegar entre os módulos, executando as atividades relativas a cada um desses módulos. A Figura 17 mostra a interface de interação usada para controle de módulo da ferramenta.

Parte do código utilizado na construção da i*GET é apresentado nos apêndices. Neles, são mostradas todas as funções e regras utilizadas na construção da ferramenta, que utiliza a linguagem CLIPS. Em especial, o APÊNDICE A mostra as regras e funções utilizadas na construção do menu inicial, apresentado na Figura 17.

```
Choose the module
by typing a number and pressing the return key.

      1.) Extracting Module.
      2.) Refining Module.
      3.) Learning Module.
      4.) Exporting Module.
      5.) Quit.
Choice: 1


###You chose: Extracting Module.###
```

Figura 17. Menu de navegação dos módulos.

Fonte: Elaborada pelo autor.

3.1 Módulo de extração

O primeiro módulo a ser apresentado é o módulo de extração. A execução do módulo é realizada em duas etapas. Na primeira etapa o conhecimento do LAL é formatado e inserido na base, sendo a segunda etapa a extração propriamente dita. A seguir é mostrado quais e como são realizadas as atividades das duas etapas.

3.1.1 Preparar o LAL

Para a utilização do motor de inferência do CLIPS, é necessária a criação de duas bases de trabalho, uma chamada "Memória de Trabalho" (base de fatos) e a outra conhecida como "Base de Regras". Para a construção da base de fatos, primeiramente é necessária a preparação do LAL. A construção da base de fatos é formatada pelo produto da primeira atividade mostrado na Figura 15. Para construção do léxico foi utilizada a ferramenta C&L, que está disponível em http://pes.inf.puc-rio.br/cel/. A ferramenta foi desenvolvida pelo grupo de Engenharia de Requisitos da PUC-Rio com o intuito de criar um ambiente colaborativo no qual, léxicos e cenários a respeito do universo do domínio da aplicação pudessem ser gerenciados, criados, editados e evoluídos. As informações são disponibilizadas de forma estruturada, em linguagem natural e simples.

Nesta etapa, o ER deve fazer o levantamento das informações que podem contribuir para elaboração do sistema. Todas as fontes de informação disponíveis devem ser pesquisadas. Neste processo, também são relevantes informações não

documentadas. Por isso, são buscadas pessoas que possuam o conhecimento tácito e que possam contribuir para o fornecimento de informações a respeito do Universo de Informação - UdI[1]. Esta é uma tarefa aberta para o ER, mas Leite (Leite et al., 2000) fornece alguns passos a serem seguidos:

- Identificar conjunto de símbolos, frases ou palavras relevantes e características do UdI.
- Classificar os símbolos em quatro categorias: sujeito, objeto, verbo e estado.
- Elaborar as descrições sobre os símbolos elicitados.
- Inspecionar as descrições do LAL.
- Validar o LAL com os atores do UdI.

Existem ainda algumas heurísticas sugeridas por (Leite et al., 2000) que devem ser seguidas para a classificação dos símbolos do LAL. A Tabela 2 mostra as regras gerais para definição dos símbolos:

Tabela 2. Regras gerais para definição de símbolos.

	Noção	Impacto
Sujeito	Quem é o sujeito.	Quais ações 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 relaciona.	Ações que podem ser aplicadas no objeto.
Estado	O que significa e quais ações levaram a esse estado.	Identificar outros estados e ações que podem ocorrer a partir do estado que se descreve.

Fonte: (Leite et al., 2000).

A Figura 18 mostra um exemplo de símbolo do LAL de cada tipo: sujeito, verbo, objeto e estado elicitados seguindo as regras gerais de definição dos símbolos.

[1] O universo de Informações (ou Universo de Discurso – UofD) inclui fontes de informação relativas ao software.

3.1.2 Estruturar o conhecimento

Após a elaboração do LAL utilizando como apoio a ferramenta C&L, o próximo passo é a estruturação das informações contidas no mesmo para inserção na base de fatos da i*GET. A estruturação é a formatação das informações do LAL para que as regras sejam capazes de extrair informações das descrições, ou seja, extrair metas concretas e flexíveis presentes nos impactos. Como pode ser visto no APÊNDICE A, a diretiva deftemplate do CLIPS é utilizada. A diretiva permite criar estruturas com atributos de tipos simples, presentes na definição da linguagem CLIPS. O APÊNDICE B mostra todas as estruturas elaboradas para o armazenamento das informações que compõem a base de fatos. Além dos templates usados na representação dos elementos do LAL, foram elaborados outros para representarem os atores, as metas concretas e flexíveis, além das ações presentes nos impactos dos símbolos do LAL.

Nome:	author
Noção:	author of **article** submitted to **conference**
Classificação:	sujeito
Impacto(s):	- sends **article** to the **conference** - receives result from **review** of the **article** - prepares **camera-ready** of the accepted **article**
Sinônimo(s):	authors.
Nome:	vote conflict
Noção:	act of deciding whether an **article** should be accepted or not.
Classificação:	verbo
Impacto(s):	- Withdraws the **article** submitted from the situation of 1 acceptance, 1 rejection and an indifferent vote
Sinônimo(s):	
Nome:	conference
Noção:	Periodic meeting of expert researchers in an research **area**
Classificação:	objeto
Impacto(s):	- researchers participate in the conference - **authors** submit **articles**
Sinônimo(s):	symposium, congress.
Nome:	conflict
Noção:	state of uncertainty about the acceptance of an **article**
Classificação:	estado
Impacto(s):	- **article** need to receive another **review**
Sinônimo(s):	conflicts.

Figura 18. Léxico Ampliado da Linguagem – Exemplos de símbolos.

Fonte: Elaborado pelo autor

A Figura 19 mostra a estrutura definida para a parametrização dos símbolos. Cada símbolo do LAL tem um identificador, nome e sua classificação (sujeito, objeto, estado ou verbo). A descrição dos impactos é fragmentada e cada impacto gera um elemento do tipo behavioralResponse, que se relaciona com o símbolo por meio do campo bhLelElementName, além de possuir um tipo e uma descrição. Na Figura 20, é mostrado um exemplo de como os atributos dos símbolos são inseridos na estrutura

apresentada. Para exemplificar, usaremos o símbolo author apresentado na Figura 18. O símbolo deu origem ao elemento elementLel e cada um dos impactos gerou um elemento behavioralResponse. Assim como apresentado no exemplo, este procedimento é realizado para todos os símbolos do domínio elicitados no LAL, estruturando as informações contidas no LAL, permitindo que as regras possam investigar o conhecimento em busca das metas.

Figura 19. Definição da estrutura de parametrização dos símbolos do LAL.

Fonte: Elaborado pelo autor.

```
(deftemplate elementLel "The LEL's element"
     (slot id         (type SYMBOL) )
     (slot name       (type STRING) )
     (slot type       (allowed-symbols subject verb state object) )
)

(deftemplate behavioralResponse "The behavioral response of the LEL element"
     (slot bhId                    (type SYMBOL) )
     (slot bhLelElementName        (type STRING) )
     (slot bhDescription           (type STRING) )
     (slot bhType                  (type SYMBOL) (allowed-symbols hardgoal softgoal) (default hardgoal))
     (slot bhProposedType          (type SYMBOL) (allowed-symbols hardgoal softgoal null) (default null))
     (slot bhCertainFactor         (type FLOAT)  (default 0.0))
     (slot bhActionId              (type SYMBOL) (default null))

)
```

Figura 20. Exemplo de símbolo do LAL estruturado para inserção na base.

```
(elementLel
      (id (gensym))
      (name "author")
      (type subject)
)
      (behavioralResponse
            (bhId (gensym))
            (bhLelElementName "author" )
            (bhDescription "sends article to the conference"))
     (behavioralResponse
            (bhId (gensym))
            (bhLelElementName "author" )
            (bhDescription "receives result from review of the article"))
     (behavioralResponse
            (bhId (gensym))
            (bhLelElementName "author" )
            (bhDescription "prepares camera-ready of the accepted article"))
```

Fonte: Elaborado pelo autor.

3.1.3 Definir AGFL - Metas dos Agentes vindas do Léxico

Após a preparação e estruturação do LAL, temos a atividade de definição da AGFL. Essa atividade pode ser dividida em três subatividades:

- Identificar Atores
- Extrair as metas dos atores a partir dos símbolos
- Refinar Metas

A Figura 21 mostra a visão geral das atividades referentes a etapa elicitação do método. Cada tarefa do módulo é mostrada com maiores detalhes a seguir.

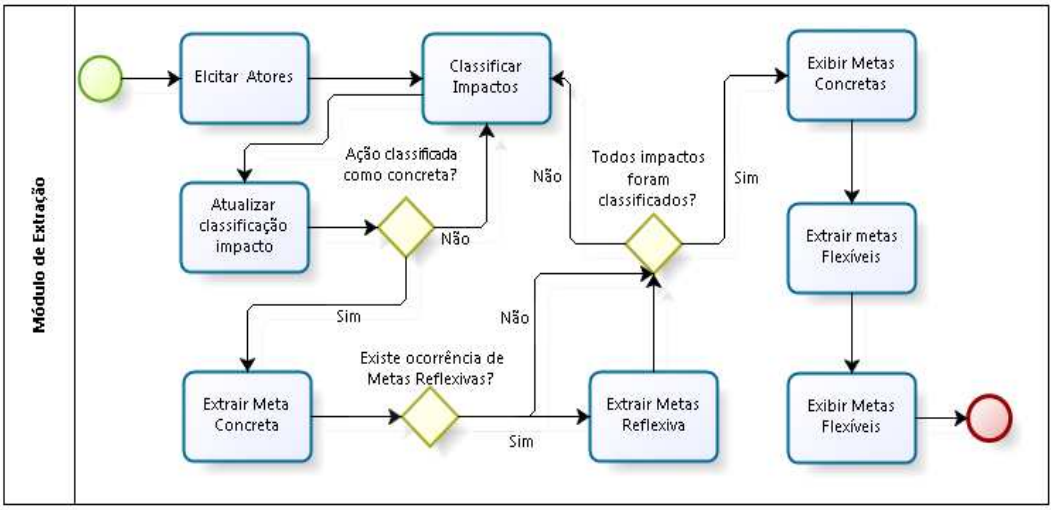


Figura 21. Fluxo de atividades do módulo de elicitação.

Fonte: Elaborado pelo autor.

3.1.3.1 Identificar Atores

Nesta tarefa, com todos os elementos do LAL já estruturados e adicionados à base de conhecimento do CLIPS juntamente com a base de regras, será iniciada a elicitação propriamente dita dos elementos. Os primeiros elementos a serem elicitados são os atores. No método, é apresentada uma heurística de identificação de atores, em que todos os símbolos classificados como sujeito no LAL e os praticantes de ações sobre os símbolos do tipo objeto são potenciais atores. Assim, foi criada uma regra segundo a qual são identificados todos os elementos classificados dessa forma. A regra ruleActorElicitation pode ser vista no APÊNDICE C.

O ator é criado com o seu nome, o mesmo utilizado pelo símbolo. Um identificador também é criado e inserido como atributo do elemento para que o ator

elicitado possa, posteriormente, ser referenciado na extração de metas por meio de seu identificador. A estrutura do elemento pode ser vista no APÊNDICE B linhas 18 a 21.

3.1.3.2 Regras para extração de metas

A fase de extração de metas é uma das fases mais complexas, devido à necessidade da experiência e sensibilidade do engenheiro de requisitos. Nesta fase, são associadas às ações presentes nos impactos dos símbolos, suas classificações segundo sua natureza.

A primeira tarefa desta atividade é a classificação dos impactos considerando as ações presentes nos mesmos. Visto que existem dois tipos de metas e metas flexíveis normalmente alteram características de metas concretas, ao classificar uma ação presente no impacto como flexível, a meta referente à ação será extraída em um momento posterior à extração de todas as metas concretas. Após a extração de todas as metas concretas, uma nova regra é disparada solicitando a elicitação das metas flexíveis.

A tarefa de classificar os elementos é de responsabilidade do ER. Para auxiliá-lo nesta tarefa, é proposto um módulo de aprendizado para a ferramenta. Esse módulo sugere classificações de ações avaliando verbos presentes nos impactos. A ferramenta apontará a possível classificação da meta considerando o histórico de decisões tomadas em outras elicitações que já utilizaram a ferramenta. Juntamente com a classificação, será indicado o Fator de Certeza (Negnevitsky, 2005) da classificação de uma determinada ação, que será avaliado segundo o número de observações, tornando a atividade de classificação das metas mais fácil. A subseção 3.3.1 fala com mais detalhe sobre o funcionamento do módulo de aprendizado da ferramenta.

Oliveira (Oliveira, 2008) sugere templates com determinados atributos para a extração dos componentes das metas. Inicialmente, apresentaremos os atributos das metas concretas, que apresentam duas estruturas de elicitação. A primeira se refere a metas extraídas a partir de símbolos do tipo sujeito. A Figura 22 mostra o formato do template sugerido. Existem alguns passos para a extração das metas:

1. Para a extração destas metas, o ator responsável já fica estabelecido a partir do símbolo.
2. O engenheiro de requisito deve informar um objeto, que deve ser um símbolo do LAL (objeto ou sujeito) já existente.

3. Um verbo na voz passiva podendo já estar presente ou não no LAL.
4. Por fim, caso a meta dependa de um segundo ator, deve ser informado um ator vindo a partir de um símbolo do tipo sujeito presente no LAL.

Algumas sugestões são dadas para a definição das metas, como a utilização do nome principal do símbolo e não seus sinônimos, além da utilização de sentenças simples e diretas. A Figura 23 mostra a interação com o ER.

Figura 22. Template para extração de metas concretas do tipo sujeito.

Fonte: (Oliveira, 2008).

No caso em que um segundo ator seja informado, existe outra heurística para

TIPO: SUJEITO	<meta concreta>				ATOR
-- impacto resposta ao por quê?	<sujeito / objeto LAL>	seja / esteja	<verbo>		<sujeito LAL>
PESQUISADORES					
-- submetem artigos.					
Porque pesquisador deseja que	artigo	seja	publicado	por	chair
Porque chair deseja que	artigo	seja	submetido	por	pesquisador
-- participam da conferência.	ação flexível				

preenchimento do que é chamado de Meta Reflexiva. A Figura 23 também mostra a ocorrência de uma Meta Reflexiva. Sua ocorrência se dá quando existe uma relação de dependência entre o depender e o dependee, ou seja, uma meta do primeiro ator (depender) depende de um segundo ator (dependee). O ciclo existe até que se tenha uma meta independente de um segundo ator ou exista uma dependência mútua entre os atores. Nesse caso, os passos indicados são:

1. Continuar até que não exista a dependência de um segundo ator ou até que exista uma dependência mútua.
2. Criar metas segundo as orientações dos passos de 1 a 5 citados anteriormente.

```
The element: author | Type: subject | BR: sends article to the conference

The historical base has classified the action in this B.R.
as [hardgoal] with degree of belief about (0.00).
Is this a softgoal or hardgoal? (softgoal: s hardgoal: h)
If there is not neither a softgoal nor a hardgoal type 'null'.
h

Because [author] want(s) <SUBJECT/OBJECT LAL> be <VERB> by <LAL SUBJECT>.
Inform the missing parameters:
article
reviewed
reviewer

In this step was identified a Reflexive Goal (RG). The RG reflects the iteraction of 2 actors.
You need to point news goals until don't have a second actor or appear a mutual dependence.
'null' will be used to represent the absence of the second actor

Because [reviewer] want(s) <SUBJECT/OBJECT LAL> be <VERB> by <LAL SUBJECT>. Inform the missing parameters:
article
submitted
author
```

Figura 23. Extração de meta concreta a partir de símbolo do tipo sujeito.

Fonte: Elaborado pelo autor.

Para metas concretas elicitadas a partir de um símbolo classificado como objeto, estado ou verbo, são exigidos quase todos os componentes das metas provenientes de símbolos classificados como sujeitos. A Figura 24 mostra o formato sugerido para a extração das metas. A diferença em relação ao formato anterior é a omissão do primeiro componente da meta. Nesse caso, é necessário o apontamento de um ator contido em um símbolo do LAL (objeto ou sujeito). Os demais atributos são os mesmo exigidos pelas metas oriundas de símbolos do tipo sujeito. A Figura 25 mostra o exemplo de extração de uma meta oriunda de um símbolo do tipo objeto.

TIPO: OBJETO	<meta concreta>				ATOR
-- impacto resposta ao por quê?	<sujeito / objeto LAL>	seja/ esteja	<verbo>		<sujeito LAL>
CONFERÊNCIA					
-- pesquisadores participam da conferência.	ação-flexível				
-- autores submetem artigos.					
Para que	artigos	sejam	revistos	por	revisor

Figura 24. Template para extração de metas concretas do tipo objeto.

Fonte: (Oliveira, 2008).

```
The element: camera-ready | Type: object | BR: prepared by the author
Is this a softgoal or hardgoal? (softgoal: s hardgoal: h)
If there is not neither a softgoal nor a hardgoal type 'null'.
h

To <SUBJECT/OBJECT LAL> be <VERB> by <LAL SUBJECT>.
Inform the missing parameters:
article
published
chair
```

Figura 25. Interação na extração de uma meta oriunda de símbolo do tipo objeto.

Fonte: Elaborado pelo autor.

Após a elicitação de todas as metas concretas, são identificadas as metas flexíveis de forma análoga. Como nas metas concretas, existe um template para auxiliar a extração dos componentes das metas flexíveis. A Figura 26 mostra o formato e os atributos sugeridos por Oliveira (Oliveira, 2008) para metas flexíveis. Para metas flexíveis, é necessária a identificação de atributos como "tipo", "tópico" e "meta concreta" associada. O campo "tipo" está relacionado a um aspecto não funcional como (e.g., Segurança, Adaptabilidade), enquanto o "tópico" refere-se ao elemento como o sistema em si ou a um elemento presente no léxico. Juntamente com o tópico e o tipo, é necessário apontar uma meta concreta, a qual a meta flexível se refere. A Figura 27 mostra a interação existente na elicitação de metas flexíveis.

TIPO: VERBO	<meta flexível>			
-- impacto resposta ao por quê?	<TIPO atributo de qualidade	[TOPICO]> sujeito/objeto LAL	<meta concreta associada>	<ator>
REVER ARTIGO				
-- é executada por revisores.	ação concreta		artigo seja revisado	revisor
-- deve ser obedecido o deadline de revisão.				
Porque	sem atraso	[revisão]	artigo seja revisado	revisor

Figura 26. Template para extração de metas flexíveis.

Fonte: (Oliveira, 2008).

```
The element – camera-ready. Type – object. BR – must respect the deadline
To <[TYPE]QUALITY ATTRIBUTE> <[TOPIC]SUBJECT/OBJECT LAL> <ASSOCIATED GOAL>.
Inform the missing parameters:
punctuality
publication
gen96
```

Figura 27. Interação na extração de metas flexíveis.

Fonte: Elaborado pelo autor.

3.2 Módulo Refinamento

O módulo de refinamento é composto por uma única atividade que envolve 4 tarefas, que são mostradas na Figura 28 e são detalhadas a seguir. No APÊNDICE D são mostradas as regras que compõem o módulo.

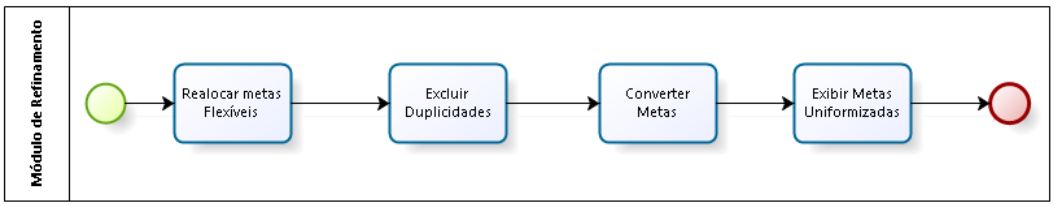


Figura 28: Fluxo de atividades do módulo de refinamento.

Fonte: Elaborado pelo autor.

3.2.1 Refinar Metas

Após a extração das metas, são necessários ajustes, pois nem todas estão no formato desejado para a construção dos modelos. No processo de elaboração, devido às descrições do LAL, podem ser elicitadas metas repetidas ou redundantes. Assim, ao final do processo de elicitação, tais metas devem ser eliminadas. A i*GET é responsável por uniformizar a redação das metas elicitadas e eliminar metas redundantes. Para a manutenção da coerência são necessárias operações onde metas flexíveis continuem relacionadas a metas concretas equivalentes as metas eliminadas devido à redundância. A primeira atividade deste módulo é a realocação de metas flexíveis. Após a realocação, as metas duplicadas são excluídas.

Retiradas as metas duplicadas, restam algumas metas originadas a partir de símbolos do tipo objeto que sofrem ajustes em suas redações, pois elas possuem formato distinto das metas extraídas a partir de símbolo do tipo sujeito. Com isso, são unificados os formatos das metas. Por fim as metas são apresentadas ao ER para que o mesmo possa validar o produto ao final da execução do módulo. A Figura 29 mostra a distinção de formatos das metas oriundas de símbolos do tipo sujeito e dos demais tipos. A Figura 30 mostra a unificação dos formatos após o refinamento das metas.

```
Goal id: gen88 | Because author want(s) article be published by chair
Goal id: gen87 | To article be reviewed by reviewer
Goal id: gen86 | To conflict be resolved by committee member
```

Figura 29. Formato das metas concretas antes do processo de refinamento.

Fonte: Elaborado pelo autor.

```
Goal id: gen83 | Beacause reviewer want(s) proposal be forwarded by chair
Goal id: gen82 | Beacause chair want(s) proposal be accepted by reviewer
Goal id: gen78 | Beacause chair want(s) article be reviewed by reviewer
```

Figura 30. Formato das metas concretas após o refinamento.

Fonte: Elaborado pelo autor.

3.3 Módulo Aprendizado

Depois de todo o processo de elicitação das metas, há um volume de informações disponível muito grande. Muitas das informações que compõem a base ao final do processo podem ser reutilizadas, agregando a expertise ao conhecimento do ER. Como consequência disso, foi elaborado para a i*GET um módulo de aprendizado. A Figura 31 mostra o fluxo de tarefas realizadas neste módulo.

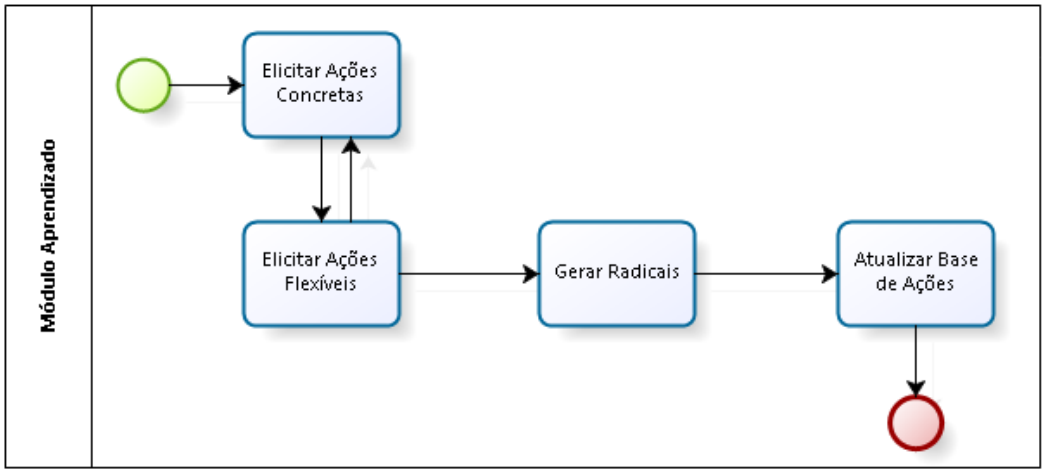


Figura 31. Fluxo de atividades do módulo de aprendizado.

Fonte: Elaborado pelo autor.

3.3.1 Capturar Aprendizado

A i*GET adiciona na base de fatos, fatos que contêm ações que representam as metas elicitadas. Este processo ocorre a cada utilização da ferramenta. Assim, nas futuras execuções, a ferramenta irá sugerir as prováveis classificações das metas presentes nas descrições dos símbolos.

Para isso, são usados os verbos e as locuções verbais presentes nos impactos. Ao final de todo o processo, são identificados os impactos que não tiveram metas identificadas automaticamente. Esses impactos são mostrados ao ER juntamente com a classificação dada durante a elicitação. Com isso, o sistema solicita a informação de qual verbo ou a locução verbal presente no impacto que representou a ação que originou a meta referente ao impacto. Assim, é gerado e adicionado à base de fatos, um fato representando a ação contida no impacto. As ações são inseridas na base, segundo a estrutura sugerida para o elemento "action" mostrada no APÊNDICE B.

Toda a criação da base de ações levantadas durante o processo é utilizada na etapa de extração das metas. Logo ao final da preparação e estruturação do LAL, tanto a base de regras quanto a base fatos são carregadas para a memória de trabalho do CLIPS. Juntamente a elas, é carregada a base de ações. Feito isso, quando é iniciado o processo de extração das metas, no primeiro módulo da ferramenta, são disparadas regras que utilizam a base de ações para classificar os impactos dos símbolos. As regras disparadas são ruleClassifyBRStem e ruleClassifyBRExpression (APÊNDICE C). Elas são responsáveis por varrer a base de fatos, verificando a ocorrência das ações nos impactos dos símbolos. Caso sejam encontradas ocorrências das ações, as regras classificam previamente os impactos segundo a classificação da ação. Assim, na extração das metas, quando o impacto de algum símbolo é mostrado para o ER, também é mostrada a classificação sugerida pela base de ações. A Figura 23 mostra como essa informação é apresentada.

Todo esse processo de busca realizado pelas regras é baseado no radical do verbo que representa a ação contida na meta. São investigados os impactos de cada símbolo em busca da ocorrência dos radicais dos verbos indicados pelo ER na extração das ações. Devido à diversidade e a dificuldade de armazenar todas as flexões de todos os verbos, é interessante reduzir as variações morfológicas das palavras.

A redução das palavras está relacionada à técnica de Normalização Morfológica (Stemming) (Barion e Lago, 2008). No processo de normalização, os prefixos, sufixos, características de gênero e grau são retirados das palavras, para obtenção do radical da palavra. O processo de Stemming é usado em sistemas de recuperação de informação. Além de muito comum, esta técnica é uma das mais efetivas em termos de consistência e aceitação dos usuários, sendo escolhida para aplicação do algoritmo de normalização de Porter (Harman, 1991).

Algoritmos do tipo Suffix Stripping possuem o benefício de uma manutenção simplificada, desde que o desenvolvedor tenha um conhecimento suficiente em linguística e morfologia. Algumas vezes, esses algoritmos são questionados devido ao desempenho em relação ao tratamento de exceções. Tais algoritmos são limitados aos léxicos que possuem sufixos bem definidos e poucas exceções. Apesar de parecer um problema, com a utilização específica de verbos em que o conjunto de sufixos é relativamente bem definido, o algoritmo apresenta um resultado satisfatório.

Depois de aplicado o processo de identificação e criação de radicais dos verbos, uma segunda etapa da fase de aprendizado é realizada. Para auxiliar o ER, é usada a avaliação de precisão das sugestões de classificação das ações. Os termos são classificados segundo seu Fator de Certeza (Negnevitsky, 2005), que é baseado no histórico de ocorrências das ações em elicitações anteriores. Essa informação também é apresentada ao ER como é mostrada na Figura 23.

3.3.2 Classificação e avaliação de sugestões

Para avaliar as sugestões é utilizado o fator de certeza. As expressões (1), (2) e (3), obtidas de Negnevitsky (Negnevitsky, 2005), mostram, respectivamente, como se obtêm a medida da crença (Measure Belief - MB), medida da descrença (Measure Disbilief – MD) e o fator de certeza (Certainty Factor - CF). Nas expressões, o índice p(H) representa a probabilidade da hipótese H ser verdadeira e p(H|E) representa a probabilidade da hipótese H ser verdadeira, dada a evidência E.

$$MB(H,E) = \begin{cases} 1 & if\ p(H) = 1 \\ \frac{\max[p(H|E),p(H)] - p(H)}{\max[1,0] - p(H)} & otherwise \end{cases} \quad (1)$$

$$MD(H,E) = \begin{cases} 1 & if\ p(H) = 0 \\ \frac{\min[p(H|E),p(H)] - p(H)}{\min[1,0] - p(H)} & otherwise \end{cases} \quad (2)$$

$$cf = \frac{MB\ (H,E) - MD(H,E)}{1 - \min[MB(H,E),MD(H,E)]} \quad (3)$$

Para exemplificar é usado o verbo votes no impacto do símbolo da Figura 32. A regra irá encontrar uma ocorrência do radical vote. Esta regra manterá o número de ocorrências do radical e o número de sugestões corretas dadas pelo sistema. Consideramos aqui sugestão correta como sendo aquela dada pelo sistema e adotada pelo ER, apresentada na etapa de extração de metas. Com os valores de sugestões corretas e ocorrências, serão calculados MB, MD e CF. Considerando que p(H) é a probabilidade de uma ação ser classificada como concreta ou flexível, uma ação tem a probabilidade de 0.5 de ser tanto concreta quanto flexível. Para p(H|E), é utilizada a probabilidade segundo os registros armazenados na base.

Para exemplificar, suponha que a ação referente ao verbo votes tenha 10 ocorrências na base da ferramenta. De 10 ocorrências no histórico da i*GET, a mesma sugeriu de forma correta, de acordo com a base histórica, a classificação da ação votes em 7 das 10 ocorrências. Logo, tem-se que:

$$MB(H \mid E) = \frac{0.7 - 0.5}{0.5} = 0.4 \qquad MD(H \mid E) = \frac{0.5 - 0.5}{0.5} = 0.0 \qquad cf = \frac{0.4 - 0.0}{1 - 0.0} = 0.4$$

O fator de certeza relacionado ao radical vote, como sendo uma meta concreta é de 0.4. Esse valor é ajustado à medida que novas execuções forem realizadas e os números de ocorrências e acertos vão sendo atualizados. Com a alteração desses valores na base, consequentemente o valor do fator de certeza para uma determinada ação vai aumentando ou diminuindo, fazendo com que as classificações fiquem mais apropriadas ou inapropriadas. Isso ocorre após a classificação de um impacto por parte do ER.

Nome:	committee member
Noção:	**researcher** of recognized competence in some **area** of the **conference**.
Classificação:	sujeito
Impacto(s):	- evaluates the **reviews** - votes for the **conflicts** resolution
Sinônimo(s):	program committee member.

Figura 32. Descrição do símbolo “committe member”.

Fonte: Elaborado pelo autor.

Todo o processo mostrado no parágrafo anterior ocorre para ações existentes na base de ações e que estejam presentes nos impactos avaliados. Caso a base não encontre nenhuma ação na descrição dos impactos avaliados, o ER a seguirá normalmente o processo de classificação, mas sem a sugestão da i*GET. Como o sistema não foi capaz de identificar uma ação, ao final do processo, a ferramenta guiará o ER pelo procedimento de criação de ação descrito no parágrafo anterior.

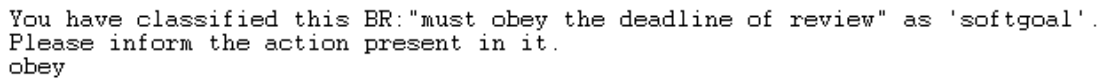


Figura 33. Tela de interação na extração de ações.

Fonte: Elaborado pelo autor.

Ao final da execução do módulo de aprendizado, as ações encontram-se na base de produção. Assim, para que a informação não seja perdida, é criado um arquivo com todas as ações extraídas no processo de elicitação e também as ações carregadas para a elicitação. Logo, quando se utiliza a i*GET para uma nova elicitação, basta que o arquivo contendo as ações seja carregado para a base de fatos do CLIPS.

3.4 Módulo de Exportação

Após a execução de todos os outros módulos da i*GET, todas as metas elicitadas ficam disponibilizadas na memória de trabalho do CLIPS. O último módulo a ser executado é responsável por disponibilizar as metas para o ER. Foi criado um módulo para exportar as metas já elicitadas, para que sejam agregadas aos demais elementos dos modelos SR e SD. A Figura 34 mostrar as atividades referentes a este módulo da ferramenta.

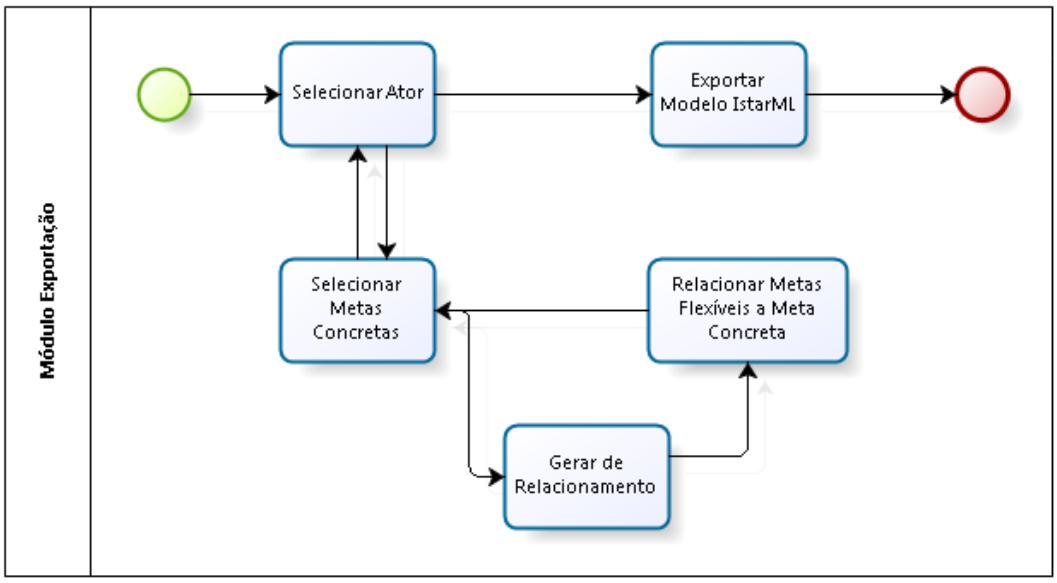


Figura 34. Fluxo de atividades do módulo de exportação.

Fonte: Elaborado pelo autor.

3.4.1 Exportar metas elicitadas

Para a tarefa de exportar as metas elicitadas é utilizada a linguagem iStarML (Cares et al., 2011). O iStarML foi proposto para representar modelos i* de diferentes meta-modelos com a finalidade de fornecer uma interoperabilidade de formatos. O iStarML foi baseado no XML (Extensible Markup Language), que é de-facto um formato de troca de informações utilizados na Internet. Segundo Cares et al. (Cares et al., 2011), existem algumas características que a tornam interessante para adoção pela comunidade:

- Por ser uma linguagem baseada em XML, a iStarML possui flexibilidade. Isso permite a especificação de estruturas opcionais.
- É uma linguagem extensível devido à capacidade de redefinição de tipos de dados XML.
- Possui uma grande capacidade de expressão devido aos atributos opcionais do XML, permitindo variações futuras da linguagem.
- É uma linguagem estável por utilizar um conjunto de conceitos estáveis nas variadas propostas relacionadas ao i*.

- É uma linguagem simples devido ao conjunto reduzido de conceitos facilitando a contribuição da linguagem.

A Tabela 3 mostra os conceitos abstratos do Framework i* e as tags representando cada um deles.

Tabela 3: Conceitos e tags do iStarML.

Conceito	Representação	Tag
Ator	Um ator representa uma entidade que pode ser uma organização, uma unidade da organização, uma pessoa ou um componente autônomo do software.	**<actor>**
Elemento Intencional	Um elemento intencional é uma entidade que permite relacionar com diferentes atores conforme a rede social e também a racionalidade interna de um ator.	**<ielement>**
Dependência	A dependência é um relacionamento no qual representa a dependência explicita de um atore (depender) em relação a outro ator (dependee).	**<dependency>** **<dependee>** **<depender>**
Limites	Um limite representa um grupo de elementos intencionais. Um tipo comum de limite é o limite dos atores, que representa a visão de um observador onipresente no âmbito do ator.	**<boundary>**
Ligação de elementos intencionais	Uma ligação de elementos intencionais representa uma ligação n-aria entre elementos intencionais (dentro ou fora dos limites do ator).	**<ielementLink>**
Ligação de associação entre atores	Uma ligação de associação entre atores é um relacionamento entre dois atores.	**<actorLink>**

Fonte: Adpatada de (Cares et al., 2011).

A Figura 35 mostra a representação de um modelo intencional i* utilizando iStarML. O diagrama está alinhado dentro da tag <istarml/>, semelhantemente os elementos intencionais do i*, são alinhados pela respectiva tag <actor/>, indicando o limite do ator, como é o caso do ator B. Dentro da fronteira, há os elementos intencionais do Framework i*. São mostrados também os elementos intencionais como as metas e tarefas. A ligação entre os elementos intencionais também pode ser vista na figura. Tanto as ligações quantos os elementos intencionais são especificados, adicionando atributos aos conceitos correspondentes nas tags de ielementLink e ielement.

Os elementos de decomposição são mostrados nas tags aninhadas das estruturas XML. A troca no valor do atributo pode estender o tipo da decomposição e até a troca do atributo muda o conjunto de relacionamentos dos elementos.

```
<istarml version="1.0">
  <diagram name="Tropos's decomposition example">
    <actor id="100" name="A">
    </actor>
    <actor id="200" name="B">
      <boundary>
        <ielement name="T1" type="task">
          <ielementLink type="decomposition" value="and">
            <ielement name="U1" type="task"/>
            <ielement name="U2" type="task"/>
            <ielement name="U3" type="task">
              <ielementLink type="decomposition" value="or">
                <ielement name="V12" type="task"/>
                <ielement name="V13" type="task"/>
              </ielementLink>
            </ielement>
          </ielementLink>
        </ielement>
      </boundary>
    </actor>
    <ielement type="goal" name="G">
      <dependency>
        <depender aref="100"/>
        <dependee aref="200"/>
      </dependency>
    </ielement>
  </diagram>
</istarml>
```

Figura 35. Exemplo de descrição de modelos usando iStarML.

Fonte: (Cares et al., 2011).

4 EXEMPLO DE APLICAÇÃO

Nesta seção, é apresentado um exemplo de elicitação de metas. Para ilustrar o funcionamento da ferramenta, será usado o modelo do sistema “Expert Committee” (EC). EC é um sistema multiagentes, aberto para suporte e gerenciamento de submissões e revisões de artigos submetidos a uma conferência ou workshop. O objetivo do sistema é assistir pesquisadores e a comissão de organização de um evento em tarefas que fazem parte do processo de revisão e submissão. O exemplo engloba todo o processo de submissão, avaliação, análise de conflito e outras atividades inerentes ao processo de realização de uma conferência.

Para extração e gestão dos símbolos, a ferramenta C&L foi usada. Com a geração dos símbolos, inicia-se o processo de extração das metas.

Seguindo todo o método já descrito, após a criação e parametrização dos símbolos, a base de dados e regras é carregada no CLIPS. As atividades realizadas foram as apresentadas na Figura 15.

```
Choose the module
by typing a number and pressing the return key.

      1.) Extracting Module.
      2.) Refining Module.
      3.) Learning Module.
      4.) Exporting Module.
      5.) Quit.
Choice: 1


###You chose: Extracting Module.###

[author] was identified as an actor.
[chair] was identified as an actor.
[committee member] was identified as an actor.
[general coordinator] was identified as an actor.
[researcher] was identified as an actor.
[reviewer] was identified as an actor.
```

Figura 36. Atores do sistema EC.

Fonte: Elaborado pelo autor.

A primeira interação foi a escolha do módulo de extração e, como primeiro resultado, são listados os atores presentes nos símbolos. Foram apontados como atores os símbolos author, chair, committee member, general coordinator, researcher e reviewer. A Figura 36 mostra a primeira interação e apresentação do resultado.

Após a identificação dos atores, inicia-se o processo de extração das metas. Em um primeiro momento, são extraídas as metas concretas. A Tabela 4 mostra todas as

metas concretas elicitadas no primeiro momento, antes do refinamento. As metas são apresentadas em ordem de inserção na base.

As metas apresentam formatos distintos, como pode ser visto na Figura 29. O formato da extração de metas concretas originadas a partir de símbolos do tipo sujeito difere das metas provenientes de símbolos que não são do tipo sujeito. Algumas metas possuem o sufixo null como elemento final. Essa é a convenção usada para representar a ausência de um atributo. No caso das metas concretas, quando não existe a dependência de um segundo ator, é utilizado esse artifício.

Com a extração das metas concretas, chega o momento de extrair as metas flexíveis. Após o apontamento das metas flexíveis e o encerramento do módulo de extração, são apresentadas as metas flexíveis. A Figura 37 mostra as metas flexíveis elicitadas e suas respectivas metas concretas associadas.

Com isso, são finalizadas as tarefas do primeiro módulo e voltamos ao menu inicial. Agora, é selecionado o módulo de refinamento da i*GET. As metas apresentadas na Tabela 4 são refinadas. O produto final obtido após esse processo é mostrado na Tabela 5. O número de metas é reduzido e as metas redundantes são excluídas. Além disso, a redação das metas é unificada.

Em seguida ao refinamento, é iniciado o módulo de aprendizado, no qual as ações dos impactos são capturadas. O APÊNDICE G mostra o arquivo gerado na íntegra. Nesse caso, todas as ações foram inseridas na base nesta elicitação. Logo, todas os elementos que representam as ações estão com os atributos actionSuccess e actionOccurrence com os mesmos valores. Esses valores são inseridos para que não haja um valor tendencioso ao fator de certeza. Assim, quando existir uma nova ocorrência da ação em outra elicitação, o fator de certeza para a sugestão será neutro, ou seja, com o valor 0.

Tabela 4: Metas concretas antes do refinamento.

Identificador	Descrição da meta
gen103	To article be submitted by author

gen102	Because author want(s) article be reviewed by reviewer
gen101	Because author want(s) article be accepted by null
gen100	To camera-ready be sent by author
gen99	Because author want(s) article be published by chair
gen98	To article be reviewed by reviewer
gen97	To conflict be decided by committee member
gen96	To article be published by chair
gen95	To article be accepted by null
gen94	Because chair want(s) article be submitted by author
gen93	To proposal be forwarded by chair
gen92	Because chair want(s) proposal be accepted by reviewer
gen91	To proposal be forwarded by chair
gen90	Because chair want(s) article be reviewed by reviewer
gen89	Because chair want(s) review be communicated by null
gen88	Because committee member want(s) conflict be resolved by null
gen87	To articles be reviewed by reviewer
gen86	To articles be received by reviewer
gen85	To conference be realized by null
gen84	To conference be realized by committee
gen83	Because general coordinator want(s) committee be compound by researcher
gen82	To review be communicated by null
gen81	Because general coordinator want(s) deadline be fixed by chair
gen80	To proposals be forwarded by chair
gen79	To article be published by chair
gen78	To proposal be approved by reviewer
gen77	To proposal be accepted by reviewer
gen76	To proposal be accepted by reviewer
gen75	To proposal be forwarded by chair
gen74	To review be communicated by null
gen73	Because researcher want(s) articles be published by chair
gen72	To article be published by chair
gen71	Because reviewer want(s) articles be reviewed by null
gen70	To conflict be resolved by committee member

```
Softgoal : Type(acknowledge) Topic[committee]
Associated Goal:article be reviewed by reviewer

Softgoal : Type(no delay) Topic[review]
Associated Goal:article be reviewed by reviewer

Softgoal : Type(quality) Topic[review]
Associated Goal:article be reviewed by reviewer

Softgoal : Type(quality) Topic[review]
Associated Goal:article be reviewed by reviewer

Softgoal : Type(honest) Topic[review]
Associated Goal:article be reviewed by reviewer

Softgoal : Type(quality) Topic[review]
Associated Goal:conflict be decided by committee member

Softgoal : Type(fair) Topic[review]
Associated Goal:conflict be decided by committee member

Softgoal : Type(punctuality) Topic[publication]
Associated Goal:article be published by chair

Softgoal : Type(quality) Topic[review]
Associated Goal:article be published by chair

Softgoal : Type(quality) Topic[article]
Associated Goal:article be published by chair

Softgoal : Type(correct) Topic[evaluation]
Associated Goal:article be accepted by null

Softgoal : Type(acknowledge) Topic[committee]
Associated Goal:conference be realized by committee
```

Figura 37. Metas flexíveis extraídas no exemplo EC.

Fonte: Elaborado pelo autor.

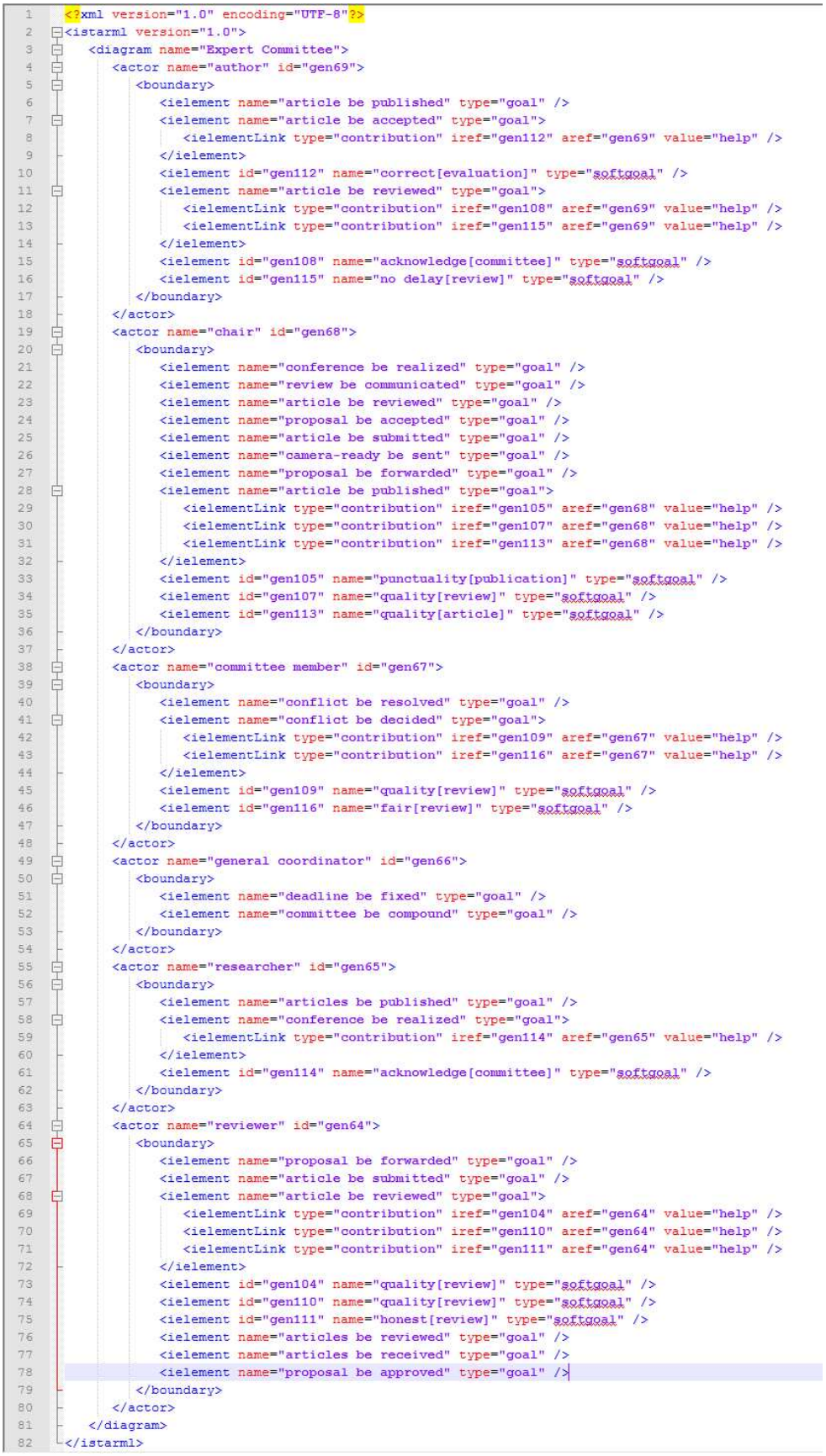
Por último, depois de toda a elicitação das metas, é hora de exportá-las. As metas são exportadas para um arquivo que é armazenado no diretório onde se encontram os demais arquivos utilizados pelo CLIPS. Após a seleção do quarto módulo, a ferramenta interage com o ER, solicitando a informação de um nome que será dado ao novo diagrama que será gerado. Com o nome informado, o fluxo segue e as metas são exportadas. A Figura 38 mostra o arquivo gerado para o exemplo mostrado.

A Figura 39 mostra as metas importadas e modeladas com a utilização do plugin jUCMNav (Jucmnav, 2013). jUCMNav foi desenvolvido para Eclipse e suporta a criação de modelos orientados a metas utilizando a linguagem GRL (Goal-oriented Requirements Language). O plugin possui a funcionalidade para importar modelos descritos em iStarML.

Tabela 5: Metas concretas após refinamento.

Identificador	Descrição da meta

gen70	committee member want(s) conflict be resolved by null
gen72	chair want(s) article be published by null
gen73	researcher want(s) articles be published by chair
gen75	chair want(s) proposal be forwarded by null
gen76	reviewer want(s) proposal be accepted by null
gen78	reviewer want(s) proposal be approved by null
gen81	general coordinator want(s) deadline be fixed by chair
gen83	general coordinator want(s) committee be compound by researcher
gen84	researcher want(s) conference be realized by committee
gen85	chair want(s) conference be realized by null
gen86	reviewer want(s) articles be received by null
gen87	reviewer want(s) articles be reviewed by null
gen89	chair want(s) review be communicated by null
gen90	chair want(s) article be reviewed by reviewer
gen92	chair want(s) proposal be accepted by reviewer
gen93	reviewer want(s) proposal be forwarded by chair
gen94	chair want(s) article be submitted by author
gen97	committee member want(s) conflict be decided by null
gen99	author want(s) article be published by chair
gen100	chair want(s) camera-ready be sent by author
gen101	author want(s) article be accepted by null
gen102	author want(s) article be reviewed by reviewer
gen103	reviewer want(s) article be submitted by author



```
<?xml version="1.0" encoding="UTF-8"?>
<istarml version="1.0">
   <diagram name="Expert Committee">
      <actor name="author" id="gen69">
         <boundary>
            <ielement name="article be published" type="goal" />
            <ielement name="article be accepted" type="goal">
               <ielementLink type="contribution" iref="gen112" aref="gen69" value="help" />
            </ielement>
            <ielement id="gen112" name="correct[evaluation]" type="softgoal" />
            <ielement name="article be reviewed" type="goal">
               <ielementLink type="contribution" iref="gen108" aref="gen69" value="help" />
               <ielementLink type="contribution" iref="gen115" aref="gen69" value="help" />
            </ielement>
            <ielement id="gen108" name="acknowledge[committee]" type="softgoal" />
            <ielement id="gen115" name="no delay[review]" type="softgoal" />
         </boundary>
      </actor>
      <actor name="chair" id="gen68">
         <boundary>
            <ielement name="conference be realized" type="goal" />
            <ielement name="review be communicated" type="goal" />
            <ielement name="article be reviewed" type="goal" />
            <ielement name="proposal be accepted" type="goal" />
            <ielement name="article be submitted" type="goal" />
            <ielement name="camera-ready be sent" type="goal" />
            <ielement name="proposal be forwarded" type="goal" />
            <ielement name="article be published" type="goal">
               <ielementLink type="contribution" iref="gen105" aref="gen68" value="help" />
               <ielementLink type="contribution" iref="gen107" aref="gen68" value="help" />
               <ielementLink type="contribution" iref="gen113" aref="gen68" value="help" />
            </ielement>
            <ielement id="gen105" name="punctuality[publication]" type="softgoal" />
            <ielement id="gen107" name="quality[review]" type="softgoal" />
            <ielement id="gen113" name="quality[article]" type="softgoal" />
         </boundary>
      </actor>
      <actor name="committee member" id="gen67">
         <boundary>
            <ielement name="conflict be resolved" type="goal" />
            <ielement name="conflict be decided" type="goal">
               <ielementLink type="contribution" iref="gen109" aref="gen67" value="help" />
               <ielementLink type="contribution" iref="gen116" aref="gen67" value="help" />
            </ielement>
            <ielement id="gen109" name="quality[review]" type="softgoal" />
            <ielement id="gen116" name="fair[review]" type="softgoal" />
         </boundary>
      </actor>
      <actor name="general coordinator" id="gen66">
         <boundary>
            <ielement name="deadline be fixed" type="goal" />
            <ielement name="committee be compound" type="goal" />
         </boundary>
      </actor>
      <actor name="researcher" id="gen65">
         <boundary>
            <ielement name="articles be published" type="goal" />
            <ielement name="conference be realized" type="goal">
               <ielementLink type="contribution" iref="gen114" aref="gen65" value="help" />
            </ielement>
            <ielement id="gen114" name="acknowledge[committee]" type="softgoal" />
         </boundary>
      </actor>
      <actor name="reviewer" id="gen64">
         <boundary>
            <ielement name="proposal be forwarded" type="goal" />
            <ielement name="article be submitted" type="goal" />
            <ielement name="article be reviewed" type="goal">
               <ielementLink type="contribution" iref="gen104" aref="gen64" value="help" />
               <ielementLink type="contribution" iref="gen110" aref="gen64" value="help" />
               <ielementLink type="contribution" iref="gen111" aref="gen64" value="help" />
            </ielement>
            <ielement id="gen104" name="quality[review]" type="softgoal" />
            <ielement id="gen110" name="quality[review]" type="softgoal" />
            <ielement id="gen111" name="honest[review]" type="softgoal" />
            <ielement name="articles be reviewed" type="goal" />
            <ielement name="articles be received" type="goal" />
            <ielement name="proposal be approved" type="goal" />
         </boundary>
      </actor>
   </diagram>
</istarml>
```

Figura 38. Metas elicitadas EC formato iStarML.

Fonte: Elaborado pelo autor.

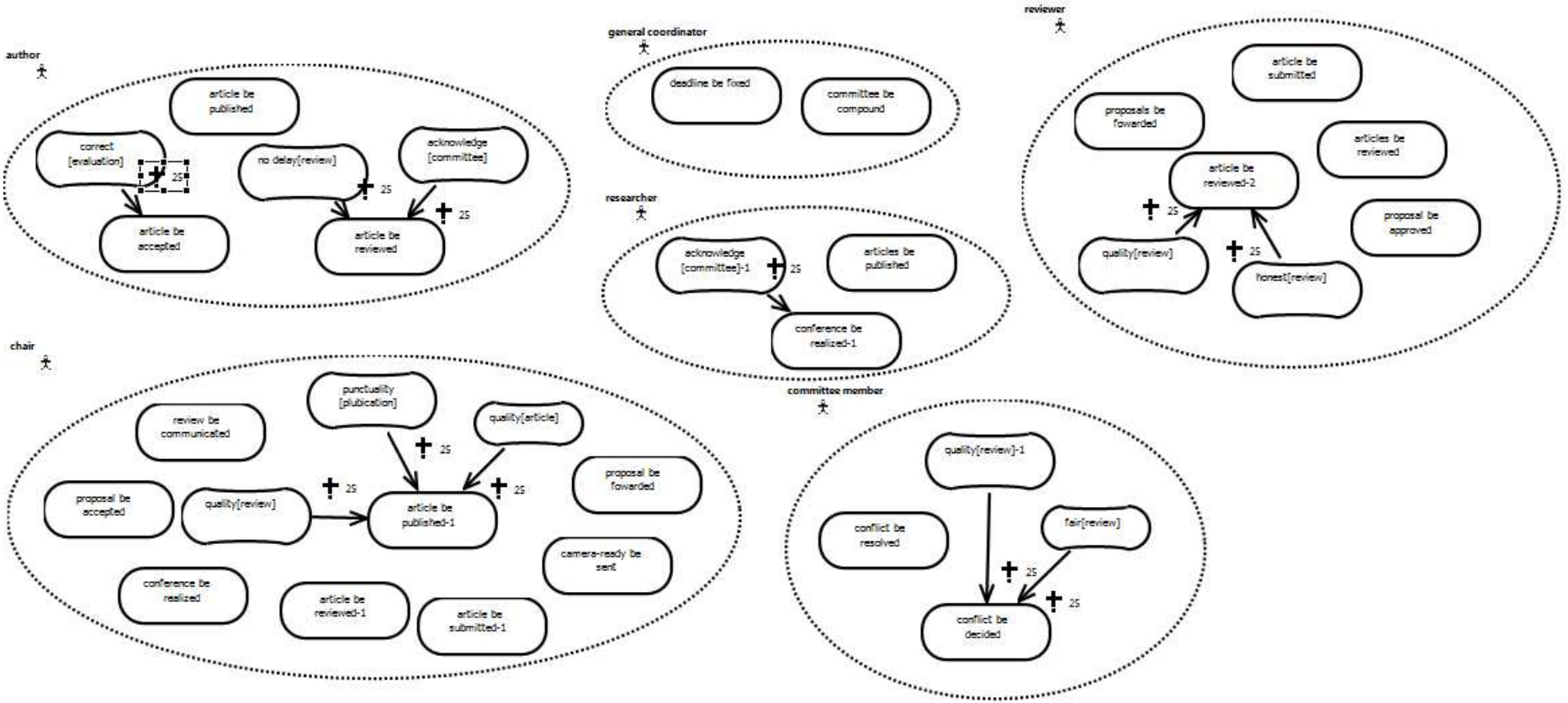


Figura 39. Representação gráfica das metas elicitadas.

Fonte: Elaborado pelo autor.

5 CONCLUSÕES

A estratégia de desenvolvimento do trabalho se mostrou satisfatória de forma a atingir os objetivos apresentados no início do trabalho.

A automação e gerenciamento da tarefa de elicitação de metas por meio da ferramenta busca melhorar a qualidade das metas a serem extraídas. A ferramenta serve de guia, orientando o ER afim de que se possa explorar de forma mais abrangente o domínio e, consequentemente, melhorando o processo de elicitação.

A maior dificuldade é a falta de trabalhos relativos à validação de modelos i*, o que implica em dificuldade na validação das metas extraídas através da i*GET. Não há uma grande gama de trabalhos que contemplem o processo de elicitação para modelos intencionais. O tratamento e validação de diagramas grandes e complexos junto aos interessados é um problema. A validação da qualidade das metas extraídas depende do conhecimento do domínio além de conceitos, técnicas e dos elementos relativos ao i* por parte do ER.

A dificuldade encontrada na validação da qualidade das metas extraídas, consequentemente gera dificuldade na validação da ferramenta. O exemplo da aplicação da ferramenta aqui apresentado, foi elaborado pelos próprios autores da pesquisa. Com os resultados obtidos, validações e o estudo de caso realizado, a adoção da ferramenta i*GET na elicitação de metas nas fases iniciais do processo de extração de requisitos, se mostrou viável. Apesar de apresentar resultados animadores, as validações em relação à ferramenta, metas e resultados obtidos foram todos feitos de maneira informal, não podendo ser consideradas para comparações. O caso de estudo serve de exemplificação no uso da ferramenta. Dessa forma, os próximos passos da pesquisa devem ser em busca de validações e resultados que possam ser formalizados na utilização da ferramenta.

Algumas características podem ser destacadas, a fim de mostrar como o trabalho apresentado se configura como um esforço preliminar em busca de contribuições maiores. Como uma das características, podemos citar uso da linguagem iStarML, que fornece interoperabilidade e uma interface de comunicação independente da ferramenta usada para modelagem, tornando-a aplicável e independente de plataforma.

Além disso, podemos citar a possibilidade de exportar as metas extraídas e utilizar outras ferramentas para atividades posteriores no processo de modelagem. Essa abordagem possibilita a utilização de ferramentas como IStar Tool (Santos, 2008), que possui verificações a respeito da integridade dos modelos através de regras Object Constraint Language – OCL (Omg, 2014).

Outro ponto positivo é a utilização de uma abordagem baseada em regras de produção, em um sistema de inferência como o CLIPS, que proporciona a obtenção de um sistema flexível e expansível, o que torna a evolução da ferramenta através de novos módulos ou a integração com outras linguagens uma possibilidade, gerando extensões à pesquisa apresentada.

5.1 Contribuições e Limitações

Podemos destacar como principais contribuições como sendo:

a) Auxiliar o ER na extração de metas para elaboração de modelos i*, oference
b) Oferecer uma ferramenta que sirva de guia para extração de metas para modelos intencionais de software.
c) Fornecer um mecanismo para geração de uma base de conhecimento de metas concretas e flexíveis.
d) Oferecer suporte ao ER na utilização do método ERi*c.

Como elemento limitante, identificamos a necessidade da validação para sistemas de grande porte e complexos. A usabilidade e aperfeiçoamento da ferramenta se darão diante da validação em problemas onde existam inúmeros objetivos e clientes. Outra limitação a ser citada é a exportação das metas elicitadas. Devido a restrições de definição da linguagem iStarML a ferramenta adiciona por padrão relacionamentos com contribuições positivas que posteriormente devem ser avaliadas pelo ER, podendo ser desenvolvidas estratégias para melhoria na definição dos relacionamentos entre as metas.

5.2 Trabalhos futuros

Algumas extensões são vislumbradas como continuação do trabalho apresentado. A seguir são apresentadas algumas sugestões de trabalhos futuros:

a) O desenvolvimento de um módulo da ferramenta capaz de automatizar a criação dos modelos completos SD e SR, seguindo o método ERi*c. Além da construção dos modelos, a ferramenta pode estender seus módulos de forma a contemplar todo o método ERi*c, como os diagramas de Painel de Intencionalidade – IP (Oliveira et al., 2008b)
b) A extensão da ferramenta com a criação de módulos capazes de investigar os modelos que estão sendo gerados a fim de validá-los junto aos interessados. Validações podem ser desenvolvidas em nível de negócio, através de simulações com os interessados, além de validações sobre a integridade dos elementos dos modelos.
c) A realização de novos estudos de caso, procurando diversificar os domínios de aplicação, para que seja possível verificar o grau de facilidade de aplicação da ferramenta. Apesar dos resultados informais sobre a eficácia da ferramenta, validações e testes feitos em ambientes controlados e com técnicas e métodos formais de verificação trarão maior credibilidade ao uso da ferramenta, além do refinamento da própria ferramenta.
d) Outro trabalho a ser desenvolvido é a integração da técnica TEKBS proposta por Lana (Lana, 2014). A técnica TEKBS pode ser incluída no fluxo de trabalho definido pela i*GET. Com a incorporação da técnica, pode ser melhorada a qualidade das metas elicitadas, o que geraria o povoamento da base de conhecimento com metas de melhor qualidade e que melhor se modelam as necessidades reais do domínio da aplicação.

APÊNDICE A

Neste primeiro apêndice, são mostradas as regras responsáveis por controle entre os módulos da ferramenta. A regra main-menu, (linha 5 a 22) é responsável por controlar e exibir o menu principal antes e após a seleção dos módulos em cada tarefa. A demais regras ruleModuleChoiceExtracting, ruleModuleChoiceRefining, ruleModuleChoiceLearning, ruleModuleChoiceExporting e ruleModuleQuit são responsáveis por controlar a seleção de cada módulo, que terá sua execução independente e poderá ser controlada pelo ER.

As regras utilizam-se da semântica de módulos oferecida pelo próprio CLIPS. Os módulos facilitam a separação semântica das regras, em que um conjunto de regras pode ser investigado separadamente e com controle de prioridade sobre outro grupo de regras.

```
1  (deffacts MAIN::init
2        (menu-level module main)
3  )
4
5  (defrule main-menu
6        ?ml <- (menu-level module main)
7  =>
8        (retract ?ml)
9        (printout t crlf)
10       (printout t
11       "Choose the module" crlf
12       "by typing a number and pressing the return key." crlf crlf
13       " 1.) Extracting Module." crlf
14       " 2.) Refining Module." crlf
15       " 3.) Learning Module." crlf
16       " 4.) Exporting Module." crlf
17       " 5.) Quit." crlf
18       "Choice: " )
19       (bind ?response (read))
20       (assert (module choice ?response))
21       (printout t crlf crlf crlf)
22 )
23
24 (defrule ruleModuleChoiceExtracting
25       (module choice 1)
26 =>
27       (printout t "###You chose: Extracting Module.###" crlf crlf)
28       (focus EXTRACTING)
29 )
30
```

```
(defrule ruleModuleChoiceRefining
      (module choice 2)
=>
      (printout t "###You chose: Refining Module.###" crlf crlf)
      (focus REFINING)
)

(defrule ruleModuleChoiceLearning
      (module choice 3)
=>
      (printout t "###You chose: Learning Module.###" crlf crlf)
      (focus LEARNING)
)

(defrule ruleModuleChoiceExporting
      (module choice 4)
=>
      (printout t "###You chose: Exporting Module.###" crlf crlf)
      (focus EXPORTING)
)

(defrule ruleModuleQuit
      (module choice 5)
=>
      (printout t "###You quit.###" crlf crlf)
      (halt)
)
```

APÊNDICE B

Nesta seção, são apresentadas as estruturas utilizadas para parametrização do conhecimento. Toda a informação do LAL deve ser convertida para que as regras possam investigar o conhecimento.

O deftemplate é uma diretiva do CLIPS utilizada para construir modelos de informação estruturada. Com ela é possível definir atributos e atribuir tipos a eles. Com isso, toda informação se torna um fatos único e atômico para a máquina de inferência do CLIPS. Abaixo, são apresentados os modelos usados na ferramenta.

```
1  (deftemplate elementLel "The LEL's element"
2        (slot id (type SYMBOL) )
3        (slot name (type STRING) )
4        (slot type (allowed-symbols subject verb state object) )
5  )
6
7  (deftemplate behavioralResponse "The behavioral "
8        (slot bhId (type SYMBOL) )
9        (slot bhLelElementName (type STRING) )
10       (slot bhDescription (type STRING) )
11       (slot bhType (type SYMBOL) (allowed-symbols hardgoal softgoal)
12             (default    hardgoal))
13       (slot bhProposedType (type SYMBOL) (allowed-symbols hardgoal
14             softgoal    null) (default null))
15       (slot bhCertainFactor (type FLOAT) (default 0.0))
16       (slot bhActionId (type SYMBOL) (default null))
17 )
```

Os templates elementLel (linha 1 a 5) e behavioralResponse (linha 7 a 17) são usados para modelar a informação dos símbolos do LAL. Eles povoam a base de conhecimento logo antes do início de todo o processo. O atributo bhDescription é o responsável por armazenar a descrição dos impactos para determinado símbolo. O template behavioralResponse possui também atributos relativos a classificação das ações presentes nos impactos dos símbolos, que são bhType e bhProposedType. Os dois atributos são usados para armazenar a classificação informada pelo ER e a classificação sugerida pelo sistema segundo a base histórica de ações, respectivamente. Já o campo bhCertainFactor armazena o fator de certeza calculado para a classificação

sugerida pela ferramenta. Por fim, o campo bhActionId é usado para armazenar o identificador da ação.

```
18  (deftemplate actorElement "Actor element"
19        (slot actorId (type SYMBOL) )
20        (slot actorName (type STRING) )
21  )
22
23  (deftemplate goalElement "Goal element"
24        (slot goalId (type SYMBOL) )
25        (slot goalElementName (type STRING) )
26        (slot goalSujectObject (type STRING) )
27        (slot goalVerb (type STRING) )
28        (slot goalSubject (type STRING) )
29        (slot goalBhId (type SYMBOL) )
30  )
31
32  (deftemplate softgoalElement "Softgoal element"
33        (slot sgId (type SYMBOL) )
34        (slot sgQualityAttribute (type STRING) )
35        (slot sgSujectObject (type STRING) )
36        (slot sgGoalId (type SYMBOL) )
37        (slot sgBhId (type SYMBOL) )
38  )
39
40  (deftemplate action "The action present in the behavioral responses"
41        (slot actionId (type SYMBOL) )
42        (slot actionStem (type STRING) (default "null"))
43        (slot actionVerb (type STRING) )
44        (slot actionType (type SYMBOL) )
45        (slot actionSuccess (type INTEGER) (default 1) )
46        (slot actionOccurrence (type INTEGER) (default 2) )
47  )
```

O template actor é usado para representar os atores elicitados no processo. O modelo é composto apenas por um identificador e um nome.

Já os templates de goalElement e softgoalElement são usados para representar o conceito de metas concreta e flexível, respectivamente. O template goalElement possui o atributo goalId que serve como identificador. O atributo goalElementName é usado para referenciar o elemento que deu origem a meta, enquanto o atributo sgBhId identifica o impacto que a ação que dá origem à meta possui. Já os atributos goalSujectObject, goalVerb, goalSubject são responsáveis por armazenar o núcleo de informações que compõem a meta concreta segundo o modelo proposto por Oliveira (Oliveira, 2008). Analogamente, o modelo softgoalElement armazena os atributos necessários para a representação das informações relacionadas a metas flexíveis.

O template action é usado para armazenar informações no processo de extração do conhecimento e classificação dos impactos. O modelo armazena informações como o radical do verbo informado e o tipo da ação representada (flexível ou concreta).

Existem também os atributos que armazenam dados sobre a ocorrência da ação e o número de vezes em que a ferramenta sugeriu, de forma correta, a classificação da ação ao ER, que são actionOccurrence e actionSuccess, respectivamente.

APÊNDICE C

Nesta seção, são apresentadas as funções e regras utilizadas no processamento dos símbolos. As regras e funções do módulo EXTRACTING são relativas à extração das metas. A seguir, são descritos os comportamentos das regras e funções.

```
1  (defmodule EXTRACTING
2        (import MAIN ?ALL)
3  )
4
5  (deffunction EXTRACTING::extractReflexiveGoals (?lelElementName ?dependeeSubject)
6        (printout t crlf "In this step was identified a Reflexive Goal (RG). The RG reflects the iteraction of 2
7        actors." crlf "You need to point news goals until don't have a second actor or appear a mutual dependence."
8        crlf "'null' will be used to represent the absence of the second actor" crlf crlf)
9
10       (while (and (not(eq ?dependeeSubject "null")) (not(eq ?dependeeSubject ?lelElementName)) )
11             (bind ?dependerSubject ?dependeeSubject)
12             (printout t "Because [" ?dependerSubject "] want(s) <SUBJECT/OBJECT LAL> be <VERB> by <LAL SUBJECT>.
13             Inform the missing parameters:"crlf)
14             (bind ?subjectObject (readline))
15             (bind ?verb (readline))
16             (bind ?dependeeSubject (readline))
17             (assert (goalElement (goalId(gensym)) (goalElementName ?dependerSubject) (goalSujectObject
18             ?subjectObject) (goalVerb ?verb) (goalSubject ?dependeeSubject)))
19       )
20 )
```

A função extractReflexiveGoals é utilizada na extração de metas concretas que possuem origem em impactos de símbolos do tipo sujeito. A função permite ao ER extrair novas metas até a ocorrência de uma dependência cíclica ou até não existir a dependência de um segundo ator na meta. A ausência de um segundo ator nesses casos é caracterizada pela palavra null. As regras foram desenvolvidas levando em consideração tal convenção.

```
(deffunction EXTRACTING::extractGoalFromElement (?lelElementType ?lelElementName ?bhId)
      (if (eq ?lelElementType subject)
            then
            (printout t crlf "Because [" ?lelElementName "] want(s) <SUBJECT/OBJECT LAL> be <VERB> by <LAL
                  SUBJECT>."crlf "Inform the missing parameters:"crlf)
            (bind ?subjectObject (readline))
            (bind ?verb (readline))
            (bind ?subject (readline))
            (assert (goalElement (goalId(gensym)) (goalElementName ?lelElementName)
                  (goalSujectObject ?subjectObject) (goalVerb ?verb) (goalSubject ?subject) (goalBhId ?bhId)))
            (if (and (neq ?lelElementName ?subject) (neq "null" ?subject))
                  then (extractReflexiveGoals ?lelElementName ?subject)
            )
      else
            (printout t crlf "To <SUBJECT/OBJECT LAL> be <VERB> by <LAL SUBJECT>. Inform the missing
                  parameters:"crlf)
            (bind ?subjectObject (readline))
            (bind ?verb (readline))
            (bind ?subject (readline))
            (assert (goalElement (goalId(gensym)) (goalElementName ?lelElementName) (goalSujectObject
                  ?subjectObject)
            (goalVerb ?verb) (goalSubject ?subject) (goalBhId ?bhId)))
      )
)

(deffunction EXTRACTING::extractSoftgoalFromElement (?bhId)

```

```
48          (printout t "To <[TYPE]QUALITY ATTRIBUTE> <[TOPIC]SUBJECT/OBJECT LAL> <ASSOCIATED GOAL> . Inform the
49          missing parameters:"crlf)
50          (bind ?qualityAttribute (readline))
51          (bind ?subjectObject (readline))
52          (bind ?goalId (readline))
53          (assert (softgoalElement (sgId(gensym)) (sgQualityAttribute ?qualityAttribute) (sgSujectObject ?subjectObject)
54          (sgGoalId ?goalId)(sgBhId ?bhId)))
55    )
```

A função extractGoalFromElement é utilizada na extração de metas concretas em ocorrências gerais. O método ERi*c propõe dois templates, um para metas provenientes de símbolos do tipo sujeito e outro para as demais classificações de símbolos. A seção 3.1.3 apresenta com detalhes a ocorrência de cada formato de meta. A função guia o ER em cada situação, fornecendo o formato esperado. Analogamente, a função extractSoftgoalFromElement apresenta para o ER o formato e atributos esperados para metas flexíveis e atributos necessários para a criação do modelo de dados softgoalElement. Além disso, a função é responsável pela inserção da meta elicitada na base de conhecimento.

```
56  (deffunction EXTRACTING::updateSuccessActionStatistic (?answer ?bhProposedType ?bhActionId)
57        (bind ?fact (nth$ 1(find-fact ((?action action)) (eq ?action:actionId ?bhActionId))))
58        (bind ?actionOccurrence (fact-slot-value ?fact actionOccurrence))
59        (bind ?actionSuccess (fact-slot-value ?fact actionSuccess))
60        (if (eq ?answer h)
61        then
62              (if (eq ?bhProposedType hardgoal)
63                    then (modify ?fact (actionSuccess (+ ?actionSuccess 1)))
64              )
65        else
66              (if (eq ?bhProposedType softgoal)
67                    then (modify ?fact (actionSuccess (+ ?actionSuccess 1)))
68              )
69        )
70        (modify ?fact (actionOccurrence (+ 1 ?actionOccurrence)))
71  )
72
```

```
(deffunction EXTRACTING::calculateCertainFactor (?actionSuccess ?actionOccurrence)
      (bind ?ph 0.5)
      (bind ?MB (/ (- (max (/ ?actionSuccess ?actionOccurrence) ?ph) ?ph)(- 1 ?ph)))
      (bind ?MD (/ (- (min (/ ?actionSuccess ?actionOccurrence) ?ph) ?ph)(- 0 ?ph)))
      (bind ?CF (/ (- ?MB ?MD) (- 1 (min ?MB ?MD))))
      (return ?CF)
)
```

As funções calculateCertainFactor e updateSuccessActionStatistic são utilizadas para o apontamento de classificações das ações exibidas para o ER. A função calculateCertainFactor é responsável por calcular o fator de certeza de um sugestão apresentada ao ER. Para isso, é necessário o número de ocorrências e sugestões corretas dadas pelo sistema. A função updateSuccessActionStatistic é responsável por manter tais atributos atualizados. São confrontadas as classificações da ferramenta e a classificação fornecida pelo ER. Assim, no caso de sucesso, os números de ocorrência e de sucesso da ação são incrementados. O valor do fator de certeza para determinada ação varia de acordo com o número de ocorrências e sugestões corretas.

```
(deffunction EXTRACTING::findSoftgoalByAGoalId (?goalId ?subjectObject ?verb ?subject)
      (bind ?start 1)
      (bind ?allSoftGoals (find-all-facts ((?softgoalElement softgoalElement)) (eq (sym-cat " "
      ?softgoalElement:sgGoalId) (sym-cat " " ?goalId))))
      (while (<= ?start (length$ ?allSoftGoals))
            (bind ?singleSoftgoal (nth$ ?start ?allSoftGoals))
            (printout t "Softgoal : Type("(fact-slot-value ?singleSoftgoal sgQualityAttribute) ") Topic[" (fact-
            slot-value ?singleSoftgoal sgSujectObject) "] | Goal:" ?subjectObject " be " ?verb " by " ?subject crlf)
            (bind ?start (+ ?start 1))
      )
)
```

A função findSoftgoalByAGoalId é utilizada para a exibição em tela de todas as metas elicitadas ao final da execução do módulo de extração. Ela busca metas flexíveis associadas a metas concretas. Assim, são exibidas para o ER as metas concretas e metas flexíveis associadas.

```
(defrule EXTRACTING::ruleActorElicitation "Point the possible actors"
      (declare (salience 100))
      (elementLel (id ?id) (name ?elementName) (type subject))
      =>
      (assert (actorElement (actorId (gensym))(actorName ?elementName)))
)

(defrule EXTRACTING::rulePrintTheActors "Print the actors elicited by the tool"
      (declare (salience 90))
      (actorElement (actorId ?actorId)(actorName ?actorName))
      =>
      (printout t "["?actorName"] was identified as an actor." crlf )
)
```

As regras rulePrintTheActors e ruleActorElicitation são responsáveis pela extração e informação dos atores. A regra ruleActorElicitation investiga todos os símbolos e identifica os possíveis atores, que serão referenciados por meio das metas em todo o restante do processo de elicitação.

```
(defrule EXTRACTING::ruleClassifyBRExpression "Classify the BR according to historical base"
      (declare (salience 80))
      ?action <- (action (actionId ?actionId) (actionStem ?actionStem) (actionType ?actionType)
            (actionSuccess ?actionSuccess) (actionOccurrence ?actionOccurrence))
      ?bh <- (behavioralResponse (bhLelElementName ?lelElementName) (bhDescription ?bhDescription)
            (bhProposedType null))
      =>
      (if (str-index ?actionStem ?bhDescription)
      then
            (modify ?bh (bhActionId ?actionId) (bhProposedType ?actionType)
            (bhCertainFactor (calculateCertainFactor ?actionSuccess ?actionOccurrence)))
      )
)

(defrule EXTRACTING::ruleClassifyBRStem "Classify the behavioral responses according to historical base"
```

```
119         (declare (salience 70))
120         ?action <- (action (actionId ?actionId) (actionStem ?actionStem) (actionType ?actionType) (actionSuccess
121         ?actionSuccess) (actionOccurrence ?actionOccurrence))
122         ?bh <- (behavioralResponse (bhLelElementName ?lelElementName) (bhDescription ?bhDescription) )
123         =>
124         (if (and (str-index ?actionStem ?bhDescription) (str-index " " ?actionStem))
125         then
126               (modify ?bh (bhActionId ?actionId) (bhProposedType ?actionType) (bhCertainFactor (calculateCertainFactor
127               ?actionSuccess ?actionOccurrence)))
128         )
129   )
130
```

As regras ruleClassifyBRExpression e ruleClassifyBRStem são responsáveis por identificar a existência de ações na base de dados e que estão presentes nos impactos dos símbolos. Caso exista a ocorrência, a regra atribui ao impacto a sugestão de classificação presente na base e que será informada ao ER no momento em que este for de fato classificar a ação presente no impacto. Além disso, a regra utiliza a função calculateCertainFactor, linhas 73 a 79, que calcula o fator de certeza da classificação sugerida ao ER.

```
131
132   (defrule EXTRACTING::ruleGoalFromSymbols "Extract goals from symbols"
133         (declare (salience 60))
134         (elementLel (name ?lelElementName) (type ?lelElementType))
135         (behavioralResponse (bhId ?bhId) (bhLelElementName ?lelElementName)
136         (bhType hardgoal) (bhDescription ?bhDescription) (bhProposedType ?bhProposedType)
137         (bhCertainFactor ?bhCertainFactor) (bhActionId ?bhActionId))
138         ?bh <- (behavioralResponse (bhLelElementName ?lelElementName) (bhDescription ?bhDescription))
139         =>
140         (printout t crlf "The element: " ?lelElementName " | Type: " ?lelElementType " | BR: " ?bhDescription crlf)
141         (if (neq ?bhActionId null)
142               then (printout t crlf "The historical base classified the action in this B.R."crlf"as ["
143               ?bhProposedType "] with degree of belief about ("(format nil "%.2f)." ?bhCertainFactor)crlf)
144         )
145         (printout t "Is this a softgoal or hardgoal? (softgoal: s hardgoal: h)"crlf"If there is not neither a softgoal
146         nor a hardgoal type 'null'." crlf)
```

```
147         (bind ?answer (read))
148         (if (neq ?bhActionId null)
149               then (updateSuccessActionStatistic ?answer ?bhProposedType ?bhActionId)
150         )
151         (if (neq ?answer null)
152               then
153               (if (eq ?answer h)
154                     then (extractGoalFromElement ?lelElementType ?lelElementName ?bhId)
155               else
156                     (printout t "Softgoals will be extracted later in the process." crlf)
157                     (modify ?bh (bhType softgoal))
158               )
159         )
160   )
```

A regra ruleGoalFromSymbols é disparada para a identificação e elicitação de metas. São investigados todos os impactos e expostos ao ER. A regra verifica a existência de um classificação prévia feita pela ferramenta nas regras ruleClassifyBRExpression e ruleClassifyBRStem . Caso exista, tanto a sugestão de classificação feita pela ferramenta quanto o fator de certeza da classificação são exibidos. Caso não exista, a regra apenas exibe o impacto a ser classificado e, posteriormente, no módulo de aprendizado, é solicitado ao ER que apresente o verbo que representa a ação contida no impacto. A regra exibe as informações ao ER e solicita a classificação definitiva. Os impactos que possuírem metas consideradas flexíveis pelo ER são apenas atualizados e extraídas posteriormente. Já as metas concretas são extraídas imediatamente por meio da função extractGoalFromElement, linhas 21 a 44.

```
161   (defrule EXTRACTING::rulePrintTheGoals "Print the hardgoals pointed by the user"
162         (declare (salience 50))
163         (goalElement (goalId ?goalId) (goalSujectObject ?subjectObject) (goalVerb ?verb) (goalSubject ?subject)
164         (goalBhId ?bhId))
165         =>
```

```
166         (if (eq ?bhId nil)
167               then
168               (printout t crlf "Goal id: " ?goalId " | To " ?subjectObject " be " ?verb " by " ?subject )
169         else
170               (bind ?fact (nth$ 1(find-fact ((?behavioralResponse behavioralResponse)) (eq ?behavioralResponse:bhId
171               ?bhId))))
172               (bind ?fact2 (nth$ 1(find-fact ((?elementLel elementLel)) (eq ?elementLel:name (fact-slot-value ?fact
173               bhLelElementName)))))
174               (if (eq (fact-slot-value ?fact2 type) subject)
175                     then
176                     (printout t crlf "Goal id: " ?goalId " | Because " (fact-slot-value ?fact2 name) " want(s) "
177                     ?subjectObject " be " ?verb " by " ?subject )
178               else
179                     (printout t crlf "Goal id: " ?goalId " | To " ?subjectObject " be " ?verb " by " ?subject )
180               )
181         )
182   )
```

A regra rulePrintTheGoals, como o próprio nome diz, exibe as metas concretas elicitadas. Após a extração de todas as metas concretas, todas elas são exibidas ao ER para que as metas flexíveis possam ser extraídas. A regra avalia se as metas elicitadas são provenientes de símbolos do tipo sujeito, já que o formato da apresentação das metas varia entre os símbolos do tipo sujeito e os demais.

```
183   (defrule EXTRACTING::ruleSoftgoalFromSymbols "Extract softgoals from symbols"
184         (declare (salience 40))
185         (elementLel (id ?elementId) (name ?lelElementName) (type ?lelElementType))
186         (behavioralResponse (bhId ?bhId) (bhLelElementName ?lelElementName) (bhDescription ?bhDescription) (bhType
187         softgoal))
188         =>
189         (printout t crlf "The element - " ?lelElementName ". Type - " ?lelElementType ". BR - " ?bhDescription crlf )
190         (extractSoftgoalFromElement ?bhId)
191   )
```

Após todas as metas concretas informadas terem sido extraídas, é hora das metas flexíveis. A regra ruleSoftgoalFromSymbols busca todos os impactos apontados pelo ER com a presença de metas flexíveis, solicitando que o ER informe os atributos para a extração da meta.

```
192 (defrule EXTRACTING::rulePrintGoals "Extract softgoals from symbols"
193       (declare (salience 20))
194       (goalElement (goalId ?goalId) (goalSujectObject ?subjectObject) (goalVerb ?verb) (goalSubject ?subject))
195       =>
196       (findSoftgoalByAGoalId ?goalId ?subjectObject ?verb ?subject)
197 )
198
199 (defrule EXTRACTING::ruleChangeModule "Change the focus to MAIN module"
200       (declare (salience 0))
201       ?fact <- (module choice 1)
202       =>
203       (assert (menu-level module main))
204       (retract ?fact)
205 )
```

Por fim, temos as regras rulePrintGoals e ruleChangeModule. A primeira exibe ao final de todo o processo de extração, as metas flexíveis associadas às metas concretas. Para isso, utiliza-se a função findSoftgoalByAGoalId, linhas 80 a 90. Já a ruleChangeModule é uma regra presente em todos os módulos e funciona como controladora de fluxo entre os módulos da ferramenta. Logo, será omitida na apresentação dos próximos módulos.

APÊNDICE D

O módulo de refinamento é responsável por excluir metas duplicadas, unificar a redação das metas, enfim, melhorar a qualidade das metas. A seguir são apresentadas as regras responsáveis por essas atividades.

```
(defmodule REFINING
      (import MAIN ?ALL)
)

(deffunction REFINING::reallocateSoftgoal (?goalId1 ?dependerSubject ?verb ?subjectObject ?dependeeSubject)
      (bind ?i 1)
      (bind ?j 1)
      (bind ?allGoals (find-all-facts ((?goalElement goalElement)) (and (eq ?goalElement:goalElementName
      ?dependerSubject) (eq  ?goalElement:goalVerb ?verb) (eq ?goalElement:goalSujectObject ?subjectObject) (eq
      ?goalElement:goalSubject ?dependeeSubject) )))
      (bind ?goalLength (length$ ?allGoals))
      (while (<= ?i ?goalLength)
            (bind ?singleGoal (nth$ ?i ?allGoals))
            (bind ?goalId (fact-slot-value ?singleGoal goalId))
            (if (neq ?goalId1 ?goalId)
                  then
                  (bind ?allSoftGoals (find-all-facts ((?softgoalElement softgoalElement)) (eq (sym-cat " "
                  ?softgoalElement:sgGoalId) (sym-cat " " ?goalId))))
                  (while (<= ?j (length$ ?allSoftGoals))
                        (bind ?singleSoftgoal (nth$ ?j ?allSoftGoals))
```

```
                        (modify ?singleSoftgoal (sgGoalId ?goalId1))
                        (bind ?j (+ ?j 1)) )
                        (retract ?singleGoal) )
                        (bind ?i (+ ?i 1)))
 )

(defrule REFINING::ruleDeleteDoubleGoal "This rule delete de double goals found in the fact base."
      (declare (salience 60))
      ?goal1 <- (goalElement (goalId ?goalId1) (goalElementName ?dependerSubject) (goalVerb ?verb) (goalSujectObject
      ?subjectObject)   (goalSubject ?dependeeSubject))
      =>
      (reallocateSoftgoal ?goalId1 ?dependerSubject ?verb ?subjectObject ?dependeeSubject)
)
```

Durante a extração, algumas metas são extraídas e ficam duplicadas e, por isso, devem ser excluídas. No entanto, podem existir metas flexíveis associadas a essas metas concretas que serão deletadas. Assim, todas as metas flexíveis são realocadas para uma mesma meta concreta equivalente por meio da função reallocateSoftgoal. A função é utilizada na regra ruleDeleteDoubleGoal, que varre todas as metas concretas para exclusão de metas redundantes.

```
(defrule REFINING::ruleConvertObjectToSubjectType "This rule is responsible for unifying the goals' structure.
      (declare (salience 40))
      (elementLel (name ?lelElementName) (type ?type))
      (goalElement (goalElementName ?lelElementName) (goalSubject ?subject))
      ?goalElement <- (goalElement (goalElementName ?lelElementName) (goalSubject ?goalSubject))
      =>
      (if (neq ?type subject)
            then
            (modify ?goalElement (goalElementName ?subject) (goalSubject "null"))
      )
)
```

Após a exclusão das metas duplicadas, ainda é necessário o ajuste da redação e o formato das metas. Como apresentado anteriormente, as metas provenientes de símbolo do tipo sujeito diferem do restante das metas na sua redação e formato. Assim, a regra ruleConvertObjectToSubjectType converte metas provenientes de símbolo que não são do tipo sujeito em metas originadas a partir de símbolos do tipo sujeito. A regra aloca o sujeito do final da meta de forma a torná-lo responsável pela meta.

```
45  (defrule REFINING::rulePrintGoals "Print the goals"
46        (declare (salience 20))
47        (goalElement (goalId ?goalId) (goalElementName ?goalElementName) (goalSujectObject ?subjectObject) (goalVerb
48        ?verb) (goalSubject ?subject) (goalBhId ?bhId))
49        =>
50        (printout t crlf "Goal id: " ?goalId " | Goal description: Because " ?goalElementName " want(s) "
51        ?subjectObject " be " ?verb " by " ?subject crlf )
52  )
53
```

APÊNDICE E

Neste apêndice, são apresentadas as regras e funções responsáveis pelo módulo de aprendizado da i*GET.

```
(defrule LEARNING::ruleClassifyNewSoftgoalAction "The rule is responsible for adding action to the knowledge base"
      (declare (salience 100))
      (behavioralResponse (bhId ?bhId) (bhDescription ?bhDescription) (bhActionId null))
      (softgoalElement (sgBhId ?bhId))
=>
      (printout t crlf "You have classified this BR:\"" ?bhDescription "\" as 'softgoal'." crlf
            "Please inform the action present in it." crlf )
      (bind ?answer (readline))
      (if (neq ?answer "null")
            then (assert (action (actionId(gensym)) (actionVerb ?answer) (actionType softgoal)))
      )
)

(defrule LEARNING::ruleClassifyNewHardgoalAction "This rule is responsible for adding action to the knowledge base"
      (declare (salience 100))
      (behavioralResponse (bhId ?bhId) (bhDescription ?bhDescription) (bhActionId null))
      (goalElement (goalBhId ?bhId))
=>
      (printout t crlf "You have classified this BR:\"" ?bhDescription "\" as 'hardgoal'." crlf "Please inform the
      action present in it." crlf )
      (bind ?answer (readline))
      (if (neq ?answer "null")
            Then (assert (action (actionId(gensym)) (actionVerb ?answer) (actionType hardgoal)))
```

```
      )

)
```

As regras ruleClassifyNewSoftgoalAction e ruleClassifyNewHardgoalAction são responsáveis por interagir com o ER na extração dos verbos que representam as metas presentes nos impactos. A duas regras funcionam de forma semelhante, exibindo o impacto e a classificação sugerida pelo próprio ER. Feito isso, é solicitado que o ER informe o verbo que represente a ação presente no impacto. Isso acontece para todos os impactos que não tiveram ocorrências registradas na base histórica. A existência de duas regras se dá devido a fato de cada uma tratar classificações distintas de ações.

```
(defrule LEARNING::ruleStemActionVerb "This rule is responsible for stemming the verbs"
      (declare (salience 80))
      ?action <- (action (actionVerb ?actionVerb) (actionStem ?actionStem))
=>
      (if (eq ?actionStem "null")
            then
            (if (str-index " " ?actionVerb)
                  then
                  (if (eq 0 (find-fact ((?template action))(eq ?template:actionStem ?actionVerb )))
                        then
                        ?action <- (modify ?action (actionStem ?actionVerb))
                  else
                        (retract ?action)
                  )
            else
                  (bind ?stem (stem ?actionVerb))
                  (if (eq 0 (length$ (find-all-facts ((?template action))(eq ?template:actionStem ?stem ))))
                        then
                              ?action <- (modify ?action (actionStem ?stem))
                  else
                        (retract ?action)
                  )
```

```
            )
        )
)
```

A regra ruleStemActionVerb é responsável por gerar Normalização Morfológica dos verbos. A regra verifica se é uma ocorrência de locução verbal ou não. Nos casos em que são informadas locuções verbais, são armazenados os verbos informados sem o processo de criação dos radicais. Já em caso de verbos simples, eles são submetidos à função actionStem. A função desencadeia os 6 passos já mencionados pelo algoritmo de Porter (seção 3.3.1). Após o final do processo, o radical obtido é armazenado no atributo stem do template action que representa a ação.

Não foi apresentado aqui o código referente ao algoritmo de Porter devido sua extensão e pelo fato de se tratar de um código de conhecimento público. Apenas vale ressaltar que todo o código foi desenvolvido utilizando a sintaxe do CLIPS.

```
(defrule LEARNING::ruleOpenActionBaseFile "This rule is responsible for creating the file containing actions and
                  their classification"
      (declare (salience 60))
=>
      (open "actionFile.clp" actionFile "w")
      (printout t "Opened file actionFile.clp." crlf)
      (printout actionFile "(deffacts MAIN::action" crlf)
      (assert (startWriteInFile))
)

(defrule LEARNING::ruleUpdateActionBaseFile "This rule is responsible for creating the file containing actions and
                  their classification"
      (declare (salience 40))
      (startWriteInFile)
      (action (actionStem ?actionStem) (actionType ?actionType) (actionSuccess ?actionSuccess) (actionOccurrence
      ?actionOccurrence)(actionVerb ?actionVerb))
=>
```

```
69          (printout actionFile "(action (actionId (gensym)) (actionStem "?actionStem") (actionVerb \""?actionVerb"\")
70          (actionType "?actionType")(actionSuccess "?actionSuccess") (actionOccurrence "?actionOccurrence"))" crlf)
71    )
72
73    (defrule LEARNING::ruleCloseActionBaseFile "This rule is responsible for creating the file containing actions and
74                        their classification"
75          (declare (salience 20))
76          ?fact<-(startWriteInFile)
77    =>
78          (retract ?fact)
79          (printout actionFile ")")
80          (close actionFile)
81          (printout t "Closed file actionFile.clp."crlf)
82    )
```

As três últimas regras presentes no módulo são ruleOpenActionBaseFile, ruleUpdateActionBaseFile e ruleCloseActionBaseFile. Todas são utilizadas na criação do arquivo responsável pela exportação das ações extraídas na elicitação. Ao finalizar uma execução, o CLIPS acaba perdendo tudo que está em sua memória. No entanto, existem informações que devem ser persistidas para utilização em novas atividades, como é o caso das ações geradas. Assim, a ferramenta cria um arquivo de forma que a informação sobre as ações sejam reutilizáveis quando a execução for finalizada. O arquivo grava as ações de forma que quando se deseja utilizar a ferramenta, apenas os arquivos gerados pela ferramenta são carregados, juntamente com a base de regras e fatos. No arquivo gerado, estarão presentes as informações sobre as classificações das ações, além de tudo que é necessário para o funcionamento da sugestão de classificação de ações fornecido pela ferramenta.

APÊNDICE F

O módulo EXPORTING é utilizado para exportar as meta elicitadas durante todo o método apresentado. Mas essas metas ainda serão trabalhadas e irão gerar modelos completos. Para facilitar as atividades do ER, a i*GET apresenta um módulo que prepara as informações levantadas na ferramenta para aproveitamento nas etapas seguintes da construção dos modelos. A ideia é que, após a execução de todos os outros módulos, o módulo de exportação seja executado. Com a execução, é criado um arquivo contendo todas as metas elicitadas na ferramenta. As metas são informadas por meio da descrição utilizando iStarML, linguagem comum para interoperabilidade de modelos i*. A seguir, é mostrado o funcionamento do módulo responsável pela exportação das informações.

```
1
2   (defmodule EXPORTING
3         (import MAIN ?ALL)
4   )
5
6   (deffunction EXPORTING::createHeaderFile (?diagramName)
7         (printout modelFile "<?xml version=\"1.0\"?> <istarml version=\"1.0\"> <diagram name=\"" ?diagramName "\">")
8   )
9
10  (deffunction EXPORTING::createFooterFile ()
11        (printout modelFile "</diagram> </istarml>")
12  )
```

As funções createHeaderFile e createFooterFile são as regras de início e fim da exportação, que geram as tags iniciais e finais de um diagrama escrito em iStarML.

```
(deffunction EXPORTING::openActorTag (?actorName ?actorId)
      (printout modelFile "<actor name=\""?actorName"\" id=\""?actorId"\">
      <boundary>")
)

(deffunction EXPORTING::closeActorTag ()
      (printout modelFile "</boundary> </actor>")
)
```

Como as funções de abertura e fechamento do arquivo, existem também duas regras responsáveis por gerar as tags para criação dos elementos atores. Cada ator possui um limite, logo, tudo que for pertencente a um determinado ator deve ser escrito entre os marcadores <boundary>, dentro do marcador <actor>. Assim, as regras geram as tags de início e fim do ator.

```
(deffunction EXPORTING::findSoftgoalByAGoalId (?goalId ?actorId)
      (bind ?start 1)
      (bind ?links "")
      (bind ?softG "")
      (bind ?allSoftGoals (find-all-facts ((?softgoalElement softgoalElement)) (eq (sym-cat " "
      ?softgoalElement:sgGoalId) (sym-cat " " ?goalId))))
      (while (<= ?start (length$ ?allSoftGoals))
            (bind ?singleSoftgoal (nth$ ?start ?allSoftGoals))
            (bind ?softG (str-cat ?softG "<ielement id =\""(fact-slot-value ?singleSoftgoal sgId)"\" name=\""(fact-
            slot-value ?singleSoftgoal sgQualityAttribute) "["(fact-slot-value ?singleSoftgoal sgSujectObject)"]\"
            type=\"softgoal\" />"))
            (bind ?start (+ ?start 1))
```

```
33              (bind ?links (str-cat ?links "<ielementLink type=\"contribution\" iref=\""(fact-slot-value
34              ?singleSoftgoal sgId)"\" aref=\""?actorId"\" value= \"help\" />"))
35  )
36        (printout modelFile ?links)
37        (printout modelFile "</ielement>")
38        (printout modelFile ?softG)
39  )
```

Após a criação de um elemento ator, mostrada na função anterior, é necessário informar quais metas estão relacionadas ao ator. A função responsável é a findSoftgoalByAGoalId. Nessa função, dado um identificador de meta concreta e um identificador de um ator, são encontradas todas as metas flexíveis relacionadas à meta concreta passada como parâmetro. Com isso, são geradas as ligações de contribuição entre as metas flexíveis e a meta concreta.

```
40  (deffunction EXPORTING::findActorGoals (?actorName ?actorId)
41        (bind ?start 1)
42        (bind ?allGoals (find-all-facts ((?goalElement goalElement)) (eq ?goalElement:goalElementName ?actorName)))
43        (while (<= ?start (length$ ?allGoals))
44              (bind ?singleGoal (nth$ ?start ?allGoals))
45              (printout modelFile "<ielement name=\""(fact-slot-value ?singleGoal goalSujectObject) " be " (fact-slot-
46              value ?singleGoal goalVerb)"\" type=\"goal\">")
47              (findSoftgoalByAGoalId (fact-slot-value ?singleGoal goalId) ?actorId)
48              (bind ?start (+ ?start 1))
49        )
50  )
```

Como a função citada anteriormente, esta função busca elementos referentes a um determinado ator. Neste caso, são buscadas as metas concretas referentes ao ator cujo identificador foi citado. Para cada meta concreta passada, todas as metas flexíveis associadas são buscadas.

```
(defrule EXPORTING::ruleOpenExportModelFile "This rule is responsible for creating the model file"
      (declare (salience 100))
=>
      (open "modelFile.istarml" modelFile "w")
      (printout t "Opened file."crlf)
      (printout t "What is the name of the diagram ?"crlf)
      (bind ?diagramName (readline))
      (createHeaderFile ?diagramName)
)

(defrule EXPORTING::ruleBodyFile "Rule responsible for creating the file header"
      (declare (salience 80))
      ?actor <- (actorElement (actorId ?actorId)(actorName ?actorName))
=>
      (openActorTag ?actorName ?actorId)
      (findActorGoals ?actorName ?actorId)
      (closeActorTag)
)

(defrule EXPORTING::ruleCloseExportModelFile "This rule is responsible for closing the model file"
      (declare (salience 60))
=>
      (createFooterFile)
      (close modelFile)
      (printout t "Close file."crlf)
)
```

As regras presentes neste módulo são utilizadas na construção do arquivo que armazena as metas e são responsáveis por fazer toda a comunicação e criação do arquivo. A primeira delas, a ruleOpenExportModelFile tem como finalidade criar o arquivo de metas, gerando toda a estrutura intera de marcações para que seja possível a importação desse arquivo posteriormente. Para isso, ela interage com ER, nomeando o diagrama e gerando o cabeçalho do arquivo.

Já a segunda regra busca os elementos interiores que povoarão o arquivo. Por meio das funções mostradas, a regra cria a tag referente ao elemento ator e associa todas suas metas. Ao final, a função closeActorTag é utilizada para finalizar a descrição do elemento ator no arquivo.

Por fim, a última regra é responsável por gerar as tags de fechamento do arquivo, que cria o rodapé do arquivo e finaliza sua criação.

APÊNDICE G

Neste apêndice, é mostrada toda a base usada na geração do exemplo de uso da ferramenta, mostrado na seção 4.

```
(deffacts MAIN::requirements "Expert committee"

(elementLel
     (id (gensym))
     (name "area of the article")
     (type object)
)
     (behavioralResponse
            (bhId (gensym))
            (bhLelElementName "area of the article" )
            (bhDescription "must be in the same area of interest of
the reviewer"))

(elementLel
     (id (gensym))
     (name "author")
     (type subject)
)
     (behavioralResponse
            (bhId (gensym))
            (bhLelElementName "author" )
            (bhDescription "sends article to the conference"))
    (behavioralResponse
            (bhId (gensym))
            (bhLelElementName "author" )
            (bhDescription "receives result from review of the
article"))
    (behavioralResponse
            (bhId (gensym))
            (bhLelElementName "author" )
            (bhDescription "prepares camera-ready of the accepted
article"))

(elementLel
     (id (gensym))
     (name "article")
     (type object)
)
     (behavioralResponse
            (bhId (gensym))
            (bhLelElementName "article" )
            (bhDescription "it is reviewed by three reviewers"))
     (behavioralResponse
            (bhId (gensym))
            (bhLelElementName "article" )
```

```
            (bhDescription "when the review process generate a
conflict, the article is reviewed by a committee member"))

(elementLel
     (id (gensym))
     (name "camera-ready")
      (type object)
)
     (behavioralResponse
            (bhId (gensym))
            (bhLelElementName "camera-ready" )
            (bhDescription "prepared by the author"))
     (behavioralResponse
            (bhId (gensym))
            (bhLelElementName "camera-ready" )
            (bhDescription "must respect the deadline"))

(elementLel
     (id (gensym))
     (name "chair")
     (type subject)
)
     (behavioralResponse
            (bhId (gensym))
            (bhLelElementName "chair" )
            (bhDescription "receives articles from the authors"))
     (behavioralResponse
            (bhId (gensym))
            (bhLelElementName "chair" )
            (bhDescription "draws up proposal for review for each
reviewer, putting name and area of the article, authors and
institution"))
    (behavioralResponse
            (bhId (gensym))
            (bhLelElementName "chair" )
            (bhDescription "sends the review proposal for each of
the reviewers,informing deadline for acceptance of the proposal and
the deadline of review of the articles"))
    (behavioralResponse
            (bhId (gensym))
            (bhLelElementName "chair" )
            (bhDescription " sends articles submitted to the event
to reviewers already defined"))
    (behavioralResponse
            (bhId (gensym))
            (bhLelElementName "chair" )
            (bhDescription "sends the review result to the author"))

(elementLel
      (id (gensym))
      (name "committee member")
      (type subject)
)
      (behavioralResponse
            (bhId (gensym))
            (bhLelElementName "committee member" )
            (bhDescription "evaluates the reviews"))
      (behavioralResponse
            (bhId (gensym))
            (bhLelElementName "committee member" )
            (bhDescription "votes for the conflicts resolution"))
```

```
107
108   (elementLel
109         (id (gensym))
110         (name "conference")
111         (type object)
112   )
113         (behavioralResponse
114                (bhId (gensym))
115                (bhLelElementName "conference" )
116                (bhDescription "researchers participate in the
117   conference"))
118        (behavioralResponse
119                (bhId (gensym))
120                (bhLelElementName "conference" )
121                (bhDescription "authors submit articles"))
122
123
124   (elementLel
125         (id (gensym))
126         (name "conflict")
127         (type state)
128   )
129         (behavioralResponse
130                (bhId (gensym))
131                (bhLelElementName "conflict" )
132                (bhDescription "article need to receive another
133   review"))
134
135   (elementLel
136         (id (gensym))
137         (name "deadline for acceptance of articles")
138         (type object)
139   )
140         (behavioralResponse
141                (bhId (gensym))
142                (bhLelElementName "deadline for acceptance of articles")
143                (bhDescription "the articles are not accepted after
144   deadline"))
145
146   (elementLel
147         (id (gensym))
148         (name "deadline of proposal")
149         (type object)
150   )
151         (behavioralResponse
152                (bhId (gensym))
153                (bhLelElementName "deadline of proposal" )
154                (bhDescription "the reviewer must respond the proposal
155   until the deadline"))
156
157   (elementLel
158         (id (gensym))
159         (name "deadline of review")
160         (type object)
161   )
162         (behavioralResponse
163                (bhId (gensym))
164                (bhLelElementName "deadline of review " )
165                (bhDescription "reviewers must respond the reviews until
166   the deadline"))
167
```

```
(elementLel
      (id (gensym))
      (name "evaluate reviews")
      (type verb)
)
      (behavioralResponse
             (bhId (gensym))
             (bhLelElementName "evaluate reviews" )
             (bhDescription "evaluates the quality of review of the
articles"))

(elementLel
      (id (gensym))
      (name "general coordinator")
      (type subject)
)
      (behavioralResponse
             (bhId (gensym))
             (bhLelElementName "general coordinator" )
             (bhDescription "invites researchers to be members of the
program committee"))
      (behavioralResponse
             (bhId (gensym))
             (bhLelElementName "general coordinator" )
             (bhDescription "decides the deadlines for receipt of
articles ,for the proposal acceptance and review of the articles"))

(elementLel
      (id (gensym))
      (name "institution")
      (type object)
)
      (behavioralResponse
             (bhId (gensym))
             (bhLelElementName "institution" )
             (bhDescription " the author and reviewer can't be
affiliated to the same institution"))

(elementLel
      (id (gensym))
      (name "list of articles")
      (type object)
)
      (behavioralResponse
             (bhId (gensym))
             (bhLelElementName "list of articles" )
             (bhDescription "prepared by the chair of the
conference"))

(elementLel
      (id (gensym))
      (name "prepare camera-ready")
      (type verb)
)
      (behavioralResponse
             (bhId (gensym))
             (bhLelElementName "prepare camera-ready" )
             (bhDescription "if not delivered, the article is not
accepted"))

(elementLel
```

```
229         (id (gensym))
230         (name "proposal")
231         (type object)
232   )
233         (behavioralResponse
234                 (bhId (gensym))
235                 (bhLelElementName "proposal" )
236                 (bhDescription "sent by the chair to reviewers"))
237       (behavioralResponse
238                 (bhId (gensym))
239                 (bhLelElementName "proposal" )
240                 (bhDescription "reviewer evaluates the proposal and
241   informs the chair if accepted or not review the articles"))
242       (behavioralResponse
243                 (bhId (gensym))
244                 (bhLelElementName "proposal" )
245                 (bhDescription "composed by the information of up to 3
246   articles"))
247       (behavioralResponse
248                 (bhId (gensym))
249                 (bhLelElementName "proposal" )
250                 (bhDescription "article that isn't accepted should be
251   relocated to another reviewer"))
252
253   (elementLel
254         (id (gensym))
255         (name "quality")
256           (type state)
257   )
258         (behavioralResponse
259                 (bhId (gensym))
260                 (bhLelElementName "quality" )
261                 (bhDescription "includes good writing, clarity of ideas,
262   good presentation and relevance of the proposal"))
263
264   (elementLel
265         (id (gensym))
266         (name "relation of reviewers")
267         (type object)
268   )
269         (behavioralResponse
270                 (bhId (gensym))
271                 (bhLelElementName "relation of reviewers" )
272                 (bhDescription "prepared by the chair of the
273   conference"))
274
275   (elementLel
276         (id (gensym))
277         (name "researcher")
278           (type subject)
279   )
280         (behavioralResponse
281                 (bhId (gensym))
282                 (bhLelElementName "researcher" )
283                 (bhDescription "submits articles"))
284       (behavioralResponse
285                 (bhId (gensym))
286                 (bhLelElementName "researcher" )
287                 (bhDescription "participates in the conference"))
288
289   (elementLel
```

```
      (id (gensym))
      (name "review article")
      (type verb)
)
      (behavioralResponse
            (bhId (gensym))
            (bhLelElementName "review article" )
            (bhDescription "executed by the reviewers"))
    (behavioralResponse
            (bhId (gensym))
            (bhLelElementName "review article" )
            (bhDescription "must obey the deadline of review"))

(elementLel
      (id (gensym))
      (name "reviewer")
      (type subject)
)
      (behavioralResponse
            (bhId (gensym))
            (bhLelElementName "reviewer" )
            (bhDescription "reviews up to 3 articles submitted to
the conference"))
    (behavioralResponse
            (bhId (gensym))
            (bhLelElementName "reviewer" )
            (bhDescription "can be invited to vote about an conflict
in articles review"))

(elementLel
      (id (gensym))
      (name "vote conflict")
      (type verb)
)
      (behavioralResponse
            (bhId (gensym))
            (bhLelElementName "vote conflict" )
            (bhDescription "Withdraws the article submitted from the
situation of 1 acceptance, 1 rejection and an indifferent vote"))
)
```

APÊNDICE H

```
1   (deffacts MAIN::action
2
3   (action (actionId (gensym)) (actionStem accept) (actionVerb "accepted")
4   (actionType hardgoal)(actionSuccess 1) (actionOccurrence 2))
5
6   (action (actionId (gensym)) (actionStem compos) (actionVerb "composed")
7   (actionType hardgoal)(actionSuccess 1) (actionOccurrence 2))
8
9   (action (actionId (gensym)) (actionStem sent) (actionVerb "sent")
10  (actionType hardgoal)(actionSuccess 1) (actionOccurrence 2))
11
12  (action (actionId (gensym)) (actionStem respond) (actionVerb "respond")
13  (actionType hardgoal)(actionSuccess 1) (actionOccurrence 2))
14
15  (action (actionId (gensym)) (actionStem review) (actionVerb "reviewed")
16  (actionType hardgoal)(actionSuccess 1) (actionOccurrence 2))
17
18  (action (actionId (gensym)) (actionStem obei) (actionVerb "obey")
19  (actionType softgoal)(actionSuccess 1) (actionOccurrence 2))
20
21  (action (actionId (gensym)) (actionStem need) (actionVerb "need")
22  (actionType softgoal)(actionSuccess 1) (actionOccurrence 2))
23
24  (action (actionId (gensym)) (actionStem particip) (actionVerb "participate")
25  (actionType softgoal)(actionSuccess 1) (actionOccurrence 2))
26
```

```
(action (actionId (gensym)) (actionStem evalu) (actionVerb "evaluates")
(actionType softgoal)(actionSuccess 1) (actionOccurrence 2))

(action (actionId (gensym)) (actionStem respect) (actionVerb "respect")
(actionType softgoal)(actionSuccess 1) (actionOccurrence 2))

(action (actionId (gensym)) (actionStem prepar) (actionVerb "prepares")
(actionType hardgoal)(actionSuccess 1) (actionOccurrence 2))

(action (actionId (gensym)) (actionStem receiv) (actionVerb "receives")
(actionType hardgoal)(actionSuccess 1) (actionOccurrence 2))

(action (actionId (gensym)) (actionStem send) (actionVerb "sends")
(actionType hardgoal)(actionSuccess 1) (actionOccurrence 2))

(action (actionId (gensym)) (actionStem vote) (actionVerb "votes")
(actionType hardgoal)(actionSuccess 1) (actionOccurrence 2))

(action (actionId (gensym)) (actionStem invit) (actionVerb "invites")
(actionType hardgoal)(actionSuccess 1) (actionOccurrence 2))

(action (actionId (gensym)) (actionStem decid) (actionVerb "decides")
(actionType hardgoal)(actionSuccess 1) (actionOccurrence 2))

(action (actionId (gensym)) (actionStem deliv) (actionVerb "delivered")
(actionType hardgoal)(actionSuccess 1) (actionOccurrence 2))

(action (actionId (gensym)) (actionStem execut) (actionVerb "executed")
(actionType hardgoal)(actionSuccess 1) (actionOccurrence 2))

(action (actionId (gensym)) (actionStem submit) (actionVerb "submitted")
(actionType hardgoal)(actionSuccess 1) (actionOccurrence 2))
)
```

REFERÊNCIAS BIBLIOGRÁFICAS

BAÍA, J. W.; BRAGA, J. L.; DE CARVALHO, L. F. Verificação de Requisitos de Transparência em Modelos iStar. 15th WER-Workshop em Engenharia de Requisitos (14th CibSE-Congresso Ibero-Americano em Engenharia de Software), 2012.

BARION, E. C. N.; LAGO, D. Mineraçao de textos. **Revista de Ciências Exatas e Tecnologia,** v. 3, n. 3, p. 123-140, 2008. ISSN 2178-6895.

BRESCIANI, P. et al. Tropos: An agent-oriented software development methodology. **Autonomous Agents and Multi-Agent Systems,** v. 8, n. 3, p. 203-236, 2004. ISSN 1387-2532.

CARES, C. et al. Towards interoperability of< i> i</i>* models using iStarML. **Computer Standards & Interfaces,** v. 33, n. 1, p. 69-79, 2011. ISSN 0920-5489.

CARVALHO, L. C. Análise de sistemas: o outro lado da informática. **Rio de Janeiro**, 1988.

CASTRO, J.; KOLP, M.; MYLOPOULOS, J. Towards Requirements-Driven Information Systems Engineering. **Information Systems Journal,** v. 27, p. 365-389, 2002.

CHUNG, L. et al. Non-Functional Requirements in Software Engineering–Kluwer Academic Publishers. **Dordrecht, USA**, 2000.

CYSNEIROS, L. M.; LEITE, J. C. S. D. P. Using the Language Extended Lexicon to Support Non-Functional Requirements Elicitation. WER, 2001, Citeseer. p.139-153.

DA SILVA, V. T.; DE LUCENA, C. J. From a conceptual framework for agents and objects to a multi-agent system modeling language. **Autonomous Agents and Multi-Agent Systems,** v. 9, n. 1-2, p. 145-189, 2004. ISSN 1387-2532.

GIARRATANO, J. C. CLIPS User's guide. **NASA Technical Report, Lyndon B Johnson Center**, 1993.

GRAU, G. et al. i* Guide. 2008. Disponível em: < http://istar.rwth-aachen.de/tiki-index.php?page_ref_id=67 >.

HARMAN, D. How effective is suffixing? **JASIS,** v. 42, n. 1, p. 7-15, 1991.

JUCMNAV. jUCMNav. 2013. Disponível em: < http://cserg.site.uottawa.ca/ucm/bin/view/ProjetSEG/WebHome >. Acesso em: 20 de Novembro de 2013.

LANA, C. A. **Técnica baseada em sinônimos para uxiliar na elaboração de modelos istar**. 2014. (Disssertação). Inédito. Departamento de Informática, Universidade Federal de Viçosa, 2014.

LEITE, J. C.; OLIVEIRA, A. A client oriented requirements baseline. Requirements Engineering, 1995., Proceedings of the Second IEEE International Symposium on, 1995, IEEE. p.108-115.

LEITE, J. C. S. D. P.; FRANCO, A. P. M. A strategy for conceptual model acquisition. Requirements Engineering, 1993., Proceedings of IEEE International Symposium on, 1993, IEEE, 1993. p.243-246.

LEITE, J. C. S. D. P. et al. A scenario construction process. **Requirements Engineering,** v. 5, n. 1, p. 38-61, 2000. ISSN 0947-3602.

LEITE, J. C. S. D. P. et al. Understanding the Strategic Actor Diagram: an Exercise of Meta Modeling. WER, 2007. p.2-12.

MARCA, D. A.; MCGOWAN, C. L. **SADT: structured analysis and design technique**. McGraw-Hill, Inc., 1987. ISBN 0070402353.

NEGNEVITSKY, M. **Artificial intelligence: a guide to intelligent systems**. Pearson Education, 2005. ISBN 0321204662.

NETO, J. Integrando Requisitos Não Funcionais ao Modelo de Objetos. **Disserteção de Mestrado, Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Informática**, 2000.

OLIVEIRA, A. D. P. A. **Engenharia de Requisitos Intencional: Um Método de Elicitação, Modelagem e Análise de Requisitos**. 2008. Tese de Doutorado, PUC-Rio

OLIVEIRA, A. D. P. A. et al. **Utilizando sistemas de conhecimento para a identificação de metas flexíveis em uma linguagem de domínio**. JAELSON CASTRO, F. A., MÁRCIA LUCENA, GILBERTO CYSNEIROS FILHO, Requirements Engineering@Brazil (ER@BR 2013), Rio de Janeiro**:** 1-6 p. 2013.

OLIVEIRA, A. D. P. A.; CYSNEIROS, L. M. Defining strategic dependency situations in requirements elicitation. 2011, The IX Workshop on Requirements Engineering; RJ, Brazil-July/2006.

OLIVEIRA, A. D. P. A. et al. Integrating scenarios, i*, and AspectT in the context of multi-agent systems. CASCON, 2006. p.204-218.

OLIVEIRA, A. D. P. A.; DO PRADO LEITE, J. C. S.; CYSNEIROS, L. M. AGFL-Agent Goals from Lexicon-Eliciting Multi-Agent Systems Intentionality. iStar, 2008a, Citeseer. p.29-32.

_____. Método ERi* c-Engenharia de Requisitos Intencional. Workshop em Engenharia de Requisitos - WER, 2008b. p.155-166

OLIVEIRA, A. D. P. A. et al. Eliciting multi-agent systems intentionality: from language extended lexicon to i* models. Chilean Society of Computer Science, 2007. SCCC'07. XXVI International Conference of the, 2007, IEEE. p.40-49.

OMG. The Object Management Group. 2014. Disponível em: < http://www.omg.org/ >. Acesso em: 20 de Fevereiro de 2014.

REGEV, G.; WEGMANN, A. Where do goals come from: the underlying principles of goal-oriented requirements engineering. Requirements Engineering, 2005. Proceedings. 13th IEEE International Conference on, 2005, IEEE. p.353-362.

SANTOS, B. **IStar Tool-Uma proposta de ferramenta para modelagem de i***. 2008. Dissertação (Mestrado). Centro de Informática, Centro de Informática, UFPE, Brasil: UFPE, Brasil

VAN LAMSWEERDE, A. Goal-oriented requirements engineering: A guided tour. Requirements Engineering, 2001. Proceedings. Fifth IEEE International Symposium on, 2001, IEEE. p.249-262.

WIKI, I. i* Wiki. 2013. Disponível em: < http://istar.rwth-aachen.de/tiki-index.php?page_ref_id=67 >. Acesso em: 20 de Dezembro de 2013.

YU, E. **Modelling Strategic Relationships for Process Reengineering**. 1995. (Ph.D.). Computer Science, University of Toronto