JOAS WESLEI BAIA

TRANSPARÊNCIA DE SOFTWARE: UMA ABORDAGEM UTILIZANDO O *FRAMEWORK* ISTAR

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
2012

Ficha catalográfica preparada pela Seção de Catalogação e Classificação da Biblioteca Central da UFV

T
B152t
2012

Baía, Joás Weslei, 1978-
Transparência de software: uma abordagem utilizando o *Framework* istar / Joás Weslei Baía. – Viçosa, MG, 2012.
101f. : il. ; 29cm.

Inclui apêndices.
Orientador: José Luís Braga.
Dissertação (mestrado) - Universidade Federal de Viçosa.
Referências bibliográficas: f. 99-101.

1. Framework (Programa de computador). 2. Engenharia de software. I. Universidade Federal de Viçosa. Departamento de Informática. Programa de Pós-Graduação em Ciência da Computação. II. Título.

CDD 22. ed. 004.25

JOAS WESLEI BAIA

TRANSPARÊNCIA DE SOFTWARE: UMA ABORDAGEM UTILIZANDO O *FRAMEWORK* ISTAR

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: 16 de outubro de 2012.

_____	_____
Alcione de Paiva Oliveira (Coorientador)	Antônio de Pádua A. Oliveira

_____

José Luís Braga
(Orientador)

LISTA DE FIGURAS

Figura 1: Grafo de interdependência dos atributos de transparência. 7
Figura 2: Representação gráfica dos elementos e conectores do iStar. 13
Figura 3: Diagrama de dependências do sistema buyer-driven e-commerce. 14
Figura 4: Diagrama de razões estratégicas do sistema buyer-driven e-commerce. .. 16
Figura 5: Estrutura de um sistema especialista baseado em regras. 20
Figura 6: Encadeamento para frente. 21
Figura 7: Encadeamento para trás. 22
Figura 8: Base de dados e conhecimento em CLIPS. 24
Figura 9: Resultado da inferência obtida com o sistema de controle de tráfego. 24
Figura 10: Atividades realizadas no método de desenvolvimento do trabalho. 25
Figura 11: Modelo de razões estratégicas do cenário adicionar foto. 30
Figura 12: Cenário adicionar foto representado parcialmente em iStarML. 31
Figura 13: Templates *element* e *elementLink*. 32
Figura 14: Template *word* usado para representar os sinônimos. 40
Figura 15: Base de fatos contendo atributos de transparência e seus sinônimos. 40
Figura 16: Modelo de razões estratégicas do domínio *Patient-Centred Care*. 44
Figura 17: Atividades do processo de identificação de atributos de transparência. .. 45

LISTA DE TABELAS

Tabela 1: Detalhamento dos atributos de transparência. 9
Tabela 2: Elementos intencionais do iStar. 11
Tabela 3: Conectores do iStar. 12
Tabela 4: Características que indicam requisitos de transparência. 28
Tabela 5: Tags iStarML. 30
Tabela 6: Mapeamento entre os elementos iStar e fatos CLIPS. 32
Tabela 7: Correspondência entre o formato istarML e fatos em CLIPS. 33
Tabela 8: Regras de produção que formam a base de conhecimento. 35
Tabela 9: Relação entre as regras de produção e os atributos de transparência. 36
Tabela 10: Descrição das regras de produção 39
Tabela 11: Relação de atributos e seus sinônimos. 41
Tabela 12: Número de ocorrência dos atributos de transparência. 47
Tabela 13: Atributos ausentes no sistema *Patient-Centred Care*. 48

LISTA DE CÓDIGOS FONTE

Código 1: A regra ruleIntentionality. 82
Código 2: A regra ruleSoftgoal. 83
Código 3: A regra ruleIntentionalityDetailed. 84
Código 4: A regra ruleAlternativeOfOperationalization. 85
Código 5: A regra ruleStrategicDependenceOfResource. 86
Código 6: A regra ruleStrategicDependenceOfGoal. 87
Código 7: A regra ruleStrategicDependenceOfSoftgoal. 88
Código 8: A regra ruleTaxonomyOfActors. 89
Código 9: A regra ruleOrganizationalStructure. 89
Código 10: A regra ruleResponsibility. 90
Código 11: A regra ruleContribution. 91
Código 12: A regra ruleDenoteSynonym. 91
Código 13: A regra ruleNotIntentionality. 92
Código 14: A regra ruleNotSofgoal 93
Código 15: A regra ruleNotDetailedIntention. 94
Código 16: A regra ruleNotAlternativeOfOperationalization. 94
Código 17: A regra ruleNotDependence. 96
Código 18: A regra ruleNotTaxonomyOfActors. 96
Código 19: A regra ruleNotOrganizationalStructure. 97
Código 20: A regra ruleNotResponsibility. 97
Código 21: A regra ruleNotContribution. 98

RESUMO

BAIA, Joas Weslei, M.Sc., Universidade Federal de Viçosa, outubro de 2012. **Transparência de Software: Uma abordagem utilizando o *Framework* iStar.** Orientador: José Luis Braga. Coorientador: Alcione de Paiva Oliveira.

Nas sociedades democráticas a demanda por informação segura e confiável baseada em transparência cresceu no panorama das transformações globais. Nesse cenário observa-se a presença de indivíduos inseridos em uma sociedade aberta, conscientes, capazes de entender e utilizar as informações disponíveis. A transparência de software é um requisito que os engenheiros de *software* precisarão demonstrar à medida que a sociedade exigir transparência, pois os sistemas de *software* permeiam as relações sociais. Nesse trabalho foram desenvolvidas estratégias para auxiliar a produção de *software* transparente. A partir de especificações de requisitos de *software* representadas com o *Framework* iStar procura-se verificar atributos de transparência que contribuem para tornar o *software* mais transparente. Essa verificação em especificações de requisitos é realizada através de regras de produção responsáveis pela identificação dos atributos de transparência. Além disso, são utilizados sinônimos desses atributos para sinalizar que a especificação de requisitos é aderente aos conceitos de transparência.

ABSTRACT

BAIA, Joas Weslei, M.Sc., Universidade Federal de Viçosa, October, 2012. **Software Transparency: An approach using Framework iStar**. Adviser: José Luis Braga. Co-Adviser: Alcione de Paiva Oliveira.

In democratic societies, the demand for reliable and secure information based on transparency has increased in the global transformation perspective. In this scenario it is noticeable the presence of individuals inserted in an open society, conscious, able to both understand and use the information available. Software transparency is a requirement that software engineers will need to present as the society demands transparency, once the software systems permeate social relations. In this work, strategies were developed to assist the production of transparent software. From software requirement specifications represented with Framework iStar, attributes of transparency are sought which contribute to make the software more transparent. Such verification of requirement specifications is carried out through production rules responsible for identifying the transparency attributes. Furthermore, synonyms of these attributes are used to indicate that the requirement specification is adherent to the transparency concepts.

SUMÁRIO

1 INTRODUÇÃO 1
1.2 Objetivo 3
1.2.1 Objetivos Específicos 3
1.3 Trabalhos Relacionados 4
2 REVISÃO BIBLIOGRÁFICA 6
2.1 Transparência de *Software* 6
2.2 Engenharia de requisitos intencional 10
2.3 O *Framework* iStar 11
2.4 Sistemas Especialistas 17
2.5 CLIPS - C *Language Integrated Production System* 23
3 MÉTODO DE DESENVOLVIMENTO DO TRABALHO 25
3.1 Modelagem de Requisitos de Software com o Framework iStar 27
3.2 Representação do Modelo de Requisitos no Formato iStarML 29
3.3 Representação do Modelo de Requisitos em Fatos Clips. 31
3.4 Representação do Conhecimento 34
3.5 Sinônimos dos Atributos de Transparência 40
4 EXEMPLO DA UTILIZAÇÃO DO SISTEMA DE VERIFICAÇÃO 42
5 CONCLUSÕES E PERSPECTIVAS FUTURAS 49
5.1 Trabalhos futuros 50
APÊNDICE 1: Base de Dados Contendo os Fatos do Sistema *Patient-Centred Care* 51
APÊNDICE 2: Base de Dados de Sinônimos dos Atributos de Transparência 54
APÊNDICE 3: Base de Conhecimento 56
APÊNDICE 4: Atributos de Transparência Identificados no Sistema *Patient-Centred Care* .. 71
APÊNDICE 5: Atributos de Transparência Ausentes no Sistema *Patient-Centred Care* 77
APÊNDICE 6: Especificação das Regras de Produção 82
REFERÊNCIAS BIBLIOGRÁFICAS 99

1 INTRODUÇÃO

Nas sociedades democráticas é crescente a demanda por informações seguras e confiáveis baseada em transparência. Esse cenário é caracterizado pela presença de indivíduos inseridos em uma sociedade aberta, que são conscientes e capazes de entender e utilizar as informações disponíveis. Nessas democracias há a preocupação com a transparência dos atos do poder público, e esforços são empregados no sentido de evitar abusos dos governantes, pois eles, no exercício do poder, praticam atos corruptos, pois a falta de transparência favorece a prática da ilegalidade. Nos regimes de governo autoritários são combatidos os princípios das sociedades abertas (HOLZNER e HOLZNER, 2003).

A transparência é importante para as organizações, elas necessitam obter informações sobre o panorama mundial. Os investidores precisam de informações sobre as oportunidades e riscos de novos mercados e com base nessas são tomadas decisões sobre as oportunidades de investimento. As organizações de proteção à saúde necessitam de fontes de informações sobre epidemias, e outras questões relativas ao bem estar dos seres humanos. Organismos internacionais necessitam de informações válidas sobre as intenções e estratégias políticas entre os países. Nesse contexto, ocorre o aumento da demanda por transparência nas relações humanas (HOLZNER e HOLZNER, 2003).

A Declaração Universal dos Direitos Humanos de 1948 garante no artigo 19 o direito à liberdade de informação, expressão e opinião em qualquer meio e independente de fronteira (ONU, 1948). A constituição da República Federativa do Brasil também traz no artigo 5º, na seção dos direitos e deveres individuais e coletivos, três incisos que resguardam a liberdade de expressão e o direito de acesso à informação. O inciso IV versa sobre a liberdade de manifestação do pensamento, proibindo apenas o anonimato. Já o inciso XIV assegura o direito de acesso à informação e o inciso LXXII versa sobre o *habeas data*, um instrumento jurídico que garante o acesso às informações armazenadas em bases de dados públicas ou de instituições de caráter público. Em seu artigo 39, a constituição também versa sobre a publicidade dos atos da Administração Pública (BRASIL, 1988).

O código de defesa do consumidor, além de garantir os direitos básicos do consumidor, também versa sobre a transparência, onde estabelece o direito de acesso a informações sobre o funcionamento dos produtos e serviços comercializados (BRASIL, 1990).

A sociedade também realiza ações para aumentar a transparência nas relações com seus representantes. A ONG (Organização não governamental) Transparência Internacional[1] é um exemplo, desde sua fundação no ano de 1993, está desenvolvendo ações no combate à corrupção e promovendo a transparência na administração pública. Ela publica anualmente o *ranking* dos países mais corruptos. No âmbito nacional, a organização Transparência Brasil[2], fundada em abril de 2000, também exerce ações semelhantes no combate à corrupção.

A Administração Pública Federal disponibilizou no ano de 2010 o Portal da Transparência[3] com o objetivo de facilitar o acesso do cidadão às informações a respeito de projetos e ações no âmbito da esfera Federal. Nota-se que o Portal da Transparência do governo brasileiro permite a transparência de acesso à informação, mas ainda não há transparência das fontes de informação, o cidadão não tem como saber se os dados disponíveis condizem com a realidade do país.

A transparência de *software* é um requisito não funcional que engenheiros precisarão demonstrar à medida que a sociedade exigir transparência nas relações com seus representantes, nas relações comerciais, sociais, enfim, nas relações humanas, pois essas relações são automatizadas pelos programas de computador. O *software* transparente possui os seguintes atributos de transparência: *acessibilidade*, *usabilidade*, *informatividade*, *entendibilidade* e *auditabilidade* (CAPPELLI e LEITE, 2008). Tais atributos são requisitos não funcionais de *software* e à medida que a demanda por transparência nas organizações crescer, o *software* deverá demonstrá-la.

Diante da importância de demonstrar a transparência nos processos de informação, propõe-se nesse trabalho desenvolver técnicas para apoiar o desenvolvimento de software transparente. A proposta é verificar a presença de

[1] http://www.transparency.org/

[2] http://www.transparencia.org.br/

[3] http://www.portaltransparencia.gov.br/

requisitos de transparência a partir de modelos de requisitos de *software* construídos com o *Framework* iStar (YU, 1995).

O iStar é um *framework* de modelagem de requisitos que permite representar as intencionalidades dos atores da organização. Metas, metas flexíveis, tarefas e recursos são organizados em torno de atores estratégicos que podem ser especializados em agentes, papéis e posições. A seção 2.3 descreve os construtores e os diagramas desse *framework.*

Este texto está organizado em cinco capítulos. O capítulo 2 apresenta a revisão bibliográfica realizada sobre as principais tecnologias utilizadas nesse trabalho. O capítulo 3 apresenta o método de desenvolvimento do trabalho, onde são discutidas as estratégias utilizadas para construir o sistema de verificação de requisitos utilizando regras de produção. No capítulo 4 é apresentado o processo de verificação de atributos de transparência através da especificação de requisitos do sistema *Patient-Centred Care*. Por fim, no capítulo 5, são apresentadas as conclusões e perspectivas futuras do trabalho desenvolvido.

1.2 Objetivo

Verificar a presença de atributos de transparência em especificações de requisitos de software representadas com o *Framewok* iStar.

1.2.1 Objetivos Específicos

- Utilizar os conceitos do *Framework* iStar para identificar atributos de transparência.
- Construir uma base de conhecimento a partir dos conceitos do *Framework* iStar para verificar a presença de atributos de transparência em especificações de requisitos.
- Representar o modelo de requisitos especificado com o iStar no formato de fatos.
- Comprovar a aplicabilidade da proposta em aplicações disponíveis na literatura.

1.3 Trabalhos Relacionados

No trabalho desenvolvido por Leal et al. (2012) foi proposta uma estrutura para promover a melhoria da transparência de *software*, onde foi apresentada uma forma de documentação de *software* fundamentada em aspectos técnicos e de uso. As informações técnicas são direcionadas aos engenheiros de *software*, tais como: arquitetura e linguagem de programação. As informações de uso são destinadas aos usuários finais do sistema, por exemplo, visão geral do *software*, suas funcionalidades e também os requisitos que não foram implementados. Essa abordagem recebeu o nome de *BuS* (Bula de *Software*) inspirada na bula de remédios que agrega informações sobre remédios.

Para organizar as informações sobre o *software* foi utilizada a tecnologia XML (*eXtensible Markup Language*). As informações técnicas e de uso são estruturadas em tópicos através da utilização de *tags*. A *tag* <*CABECALHOSOFTWARE/*> descreve a seção do documento contendo o nome do software, sua composição e seus objetivos. Na *tag* <*INFORMACAOGERALCIDADAO/*> ficam estruturadas as informações sobre o uso do sistema e sobre o seu escopo. Por fim, na *tag* <*INFORMACAOTECNICA/*> são disponibilizadas informações sobre a tecnologia envolvida no desenvolvimento do *software* e que tem como público alvo os engenheiros de *software*.

A documentação de *software BuS* visa potencializar a transparência contribuindo para que ele seja aderente aos atributos de transparência *uniformidade, amigabilidade, simplicidade, intuitividade, operabilidade* e *clareza. A* abordagem apresentada nesse texto procura indícios da presença de atributos de transpârencia em especificações de requisitos iStar. As duas estratégias possuem objetivos semelhantes, ambas buscam tornar o *software* mais transparente.

O trabalho desenvolvido por Kiyavitskaya et al. (2007) aborda a verificação de requisitos a partir de processamento de linguagem natural com o objetivo de identificar ambiguidades em especificações de requisitos. Os requisitos representados em linguagem natural são submetidos ao procedimento de medição de ambiguidades, onde as sentenças potencialmente ambíguas são verificadas. Além disso, é identificado o motivo que tornaria a sentença ambígua auxiliando o

trabalho do engenheiro de *software* na remoção de tais ambiguidades. A abordagem de verificação de atributos de transparência desenvolvida nesse trabalho não utiliza processamento de linguagem natural, porém procura identificar sinônimos desses atributos de transparência nos modelos de requisitos.

No trabalho de Fuxman et al. (2001) a metodologia Tropos é empregada para verificar os aspectos dinâmicos de dependência entre atores estratégicos. O modelo de especificação de requisitos iStar é representado na linguagem *Formal Tropos* onde atores, metas, tarefas, recursos e dependências são declarados juntamente com as restrições usadas para verificar o modelo. O método de verificação de atributos de transparência difere dessa abordagem, pois o modelo a ser verificado é organizado em uma base de dados enquanto o conhecimento usado para a verificação é implementado nas regras de produção. Há dois módulos distintos para realizar a verificação: a base de fatos e a base de regras. A base de fatos é composta pela especificação de requisitos, enquanto a base de regras é formada pelas regras de produção que são aplicadas à base de fatos para inferir a presença ou ausência dos atributos de transparência.

A estratégia utilizada no trabalho de Duran et al. (2001) emprega a tecnologia XML/XSLT para verificar especificações de requisitos sob os aspectos de ambiguidade, completude e rastreabilidade. A verificação dos requisitos é realizada através da aplicação de folhas de estilos sobre o modelo de requisitos representado em XML. Enquanto a abordagem aqui apresentada utiliza regras de produção para verificar se na especificação de requisitos representada com o *framework* iStar há indícios de atributos de transparência.

A diferença entre a estratégia utilizada nesse trabalho e as abordagens utilizadas por Duran et al. (2001), Fuxman et al. (2001) e Kiyavitskaya et al. (2007) está no objetivo da verificação. Ela busca identificar atributos de transparência em especificações de requisitos, enquanto as demais procuram verificar se essas especificações estão em conformidade com as necessidades e interesses dos envolvidos no ambiente organizacional onde o sistema irá operar. A transparência de *software* será mais uma entre as expectativas dos *stakeholders* à medida que ela for exigida pelos diversos segmentos da sociedade.

2 REVISÃO BIBLIOGRÁFICA

2.1 Transparência de *Software*

A transparência de software tem sido explorada como um requisito não funcional. A Figura 1 mostra o grafo de interdependência dos atributos de transparência. Esses atributos foram reunidos em cinco grupos: *auditabilidade*, *entendibilidade*, *informatividade*, *usabilidade* e *acessibilidade*. Cada um desses grupos reúne atributos que contribuem para a obtenção da transparência. Por exemplo, *auditabilidade* agrega os atributos de qualidade *validação*, *controlabilidade*, *verificabilidade*, *rastreabilidade* e *explicabilidade* (CAPPELLI et al., 2007).

Considerando o aspecto social, o conceito de transparência está relacionado ao príncipio democrático de ser informado e ter acesso à informação, onde os cidadãos desejam obter maiores informações sobre fatos e processos (HOLZNER e HOLZNER, 2003).

Nos processos organizacionais a transparência é definida como a existência de processos que permitem aos indivíduos obterem informações sobre a organização através do acesso, uso, apresentação, entendimento e auditabilidade (CAPPELLI, 2009).

A transparência de software é um requisito não funcional, pois é caracterizada por atributos de qualidade, podendo ser julgada de forma diferente de acordo com o a visão do indivíduo, dessa forma ela é considerada uma meta-flexível, ou um *softgoal*, ou seja, o *software* pode ser considerado transparente do ponto de vista de uma pessoa, enquanto pode não ser transparente ou ainda menos transparente para outra (LEAL et al., 2011). O *Software* transparente deve possuir os atributos de *acessibilidade*, *usabilidade*, *informatividade*, *entendibilidade* e *auditabilidade*, pois nos processos organizacionais transparentes esses atributos devem estar presentes, e considerando que os sistemas de *software* automatizam tais processos, eles devem incorporá-los (OLIVEIRA et al., 2007).

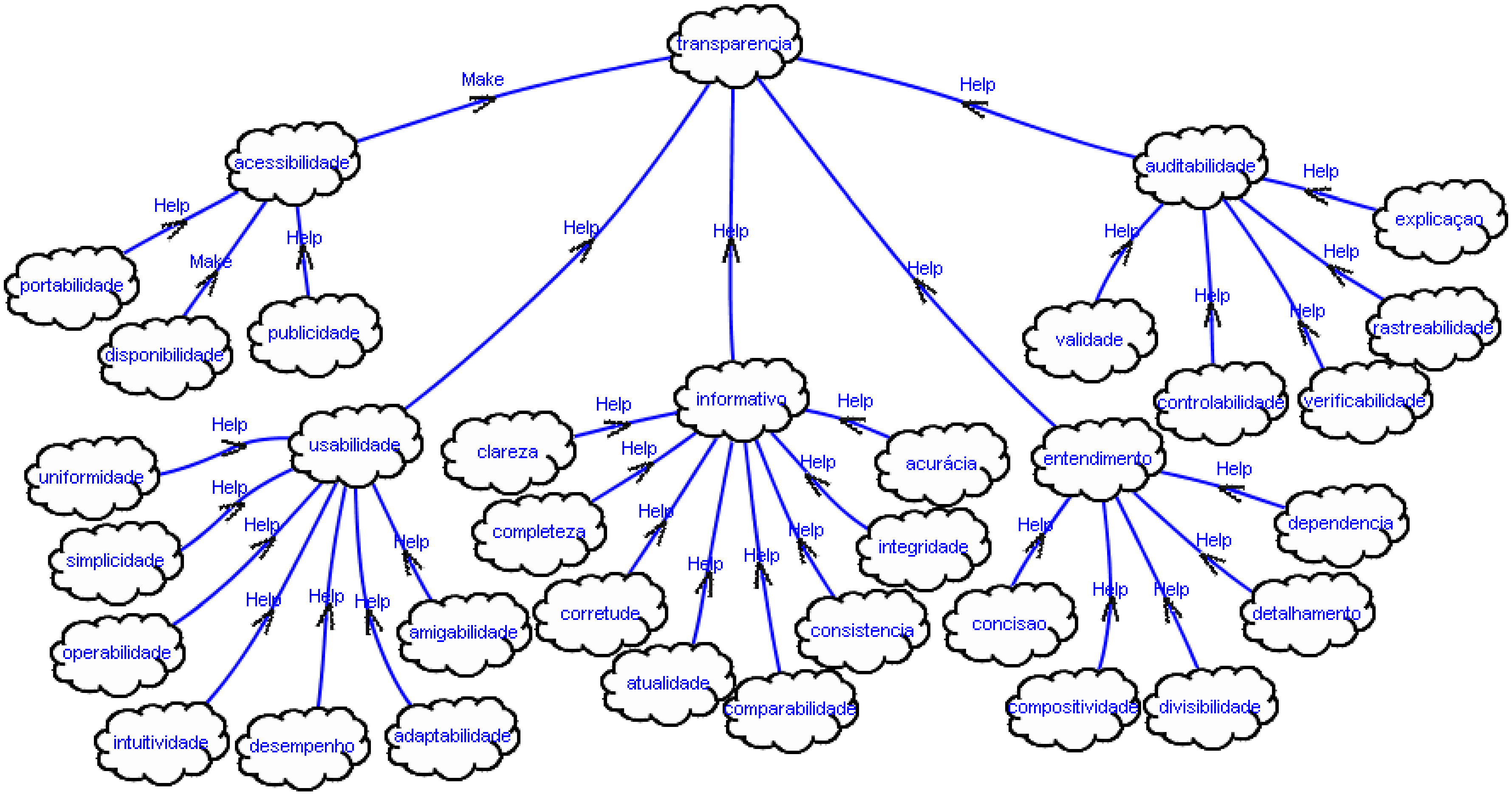

Figura 1: Grafo de interdependência dos atributos de transparência.

Fonte: Capelli, 2009.

No grafo da Figura 1, os rótulos dos arcos indicam a contribuição dos atributos para a transparência. O arco que conecta os vértices *portabilidade* e *acessibilidade* está rotulado com a palavra *Help,* indicando que o atributo de origem contribui parcialmente para que o atributo *acessibilidade* seja alcançada. Outro rótulo presente nessa figura é o *Make* utilizado para indicar que o vértice de origem contribui suficientemente para que o vértice destino seja alcançado. Ele foi utilizado para sinalizar a contribuição entre os vértices *disponibilidade* e *acessibilidade* e desse último com o vértice *transparência*. A Tabela 3 apresentada na seção 2.3 exibe a lista dos conectores de contribuição disponíveis no *Framework* iStar.

A Tabela 1 apresenta os atributos de transparência organizados segundo suas correlações, além disso, é exibida a definição de cada atributo de transparência. Por exemplo, o atributo de transparência *disponiblidade* é definido como a capacidade de ser utilizado no momento em que se fizer necessário e está classificado no grupo de atributos que contribuem para o grau de transparência *Acessibilidade* (CAPPELLI, 2009).

Os atributos de transparência exibidos na Tabela 1 são os vértices do grafo da Figura 1. Por exemplo, os atributos listados na coluna dois são os vértices folha desse grafo e os atributos exibidos na primeira coluna formam, a partir da raiz, o próximo nível de profundidade da árvore.

Nesse trabalho a especificação de requisitos representada com o *Framework* iStar é verificada em busca da presença dos atributos de transparência listados na Tabela 1. Essa abordagem busca mostrar a aderência dos requisitos de *software* com os atributos de transparência.

Tabela 1: Detalhamento dos atributos de transparência.
Fonte: Adaptado de Cappelli (2009).

	Atributos	Descrição do atributo
Acessibilidade	Portabilidade	Capacidade de ser usado em diferentes ambientes.
	Disponibilidade	Capacidade de ser utilizado no momento necessário.
	Publicidade	Capacidade de ser apresentado.
Usabilidade	Uniformidade	Capacidade de manter uma única forma.
	Simplicidade	Capacidade de não apresentar dificuldades ou obstáculos.
	Operabilidade	Capacidade de ser operacional.
	Intuitividade	Capacidade de ser utilizado sem aprendizado prévio.
	Desempenho	Capacidade de operar adequadamento.
	Adaptabilidade	Capacidade de mudar de acordo com as circunstâncias.
	Amigabilidade	Capacidade de utilização sem esforço.
Informatividade	Clareza	Capacidade de nitidez e compreensão.
	Completeza	Capacidade de não faltar nada do que pode ou deve ter.
	Corretude	Capacidade de ser isento de erros.
	Atualidade	Capacidade de estar no estado atual.
	Comparabilidade	Capacidade de ser comparado.
	Consistência	Capacidade de resultado aproximado de várias medições de um mesmo item.
	Integridade	Capacidade de correto e imparcial.
	Acurácia	Capacidade de execução isenta de erros sistemáticos.
Entendibilidade	Concisão	Capacidade de ser resumido.
	Compositividade	Capacidade de construir ou formar a partir de diferentes pares.
	Divisibilidade	Capacidade de ser particionado.
	Detalhamento	Capacidade de descrever em minúcias.
	Dependência	Capacidade de identificar a relação entre as partes de um todo.
Auditabilidade	Validável	Capacidade de ser testado por experimento ou observação para identificar se o que está sendo feito é correto.
	Controlabilidade	Capacidade de domínio.
	Verificabilidade	Capacidade de identificar se o que está sendo feito é o que deve ser feito.
	Rastreabilidade	Capacidade de seguir o desenvolvimento de um processo ou a construção de uma informação, suas mudanças e justificativas.
	Explicável	Capacidade de informar a razão de algo.

2.2 Engenharia de requisitos intencional

A engenharia de requisitos trabalha na elicitação, especificação, análise, documentação e evolução dos serviços e restrições que devem ser implementados nos sistemas de *software*. A engenharia de requisitos baseada em objetivos, além de contemplar essas fases, tem como objetivo a análise de requisitos com base nas intenções dos indivíduos presentes no domínio do problema. A identificação correta dos desejos dos atores possibilita que o *software* seja desenvolvido de acordo com as expectativas e interesses dos *stakeholders* (LAPOUCHNIAN, 2005).

A intencionalidade é a representação das motivações e desejos dos *stakeholders*. Ela é um ponto importante para a transparência, pois através dela é possível obter rastreabilidade e verificabilidade das ações realizadas pelos atores para atingirem seus objetivos (LEITE e CAPPELLI, 2008).

O iStar é um *framework* de modelagem de requisitos proposto por Eric Yu (1995) que permite representar intencionalidade dos atores envolvidos no ambiente no qual o *software* irá operar. Ele oferece ao projetista um conjunto de construtores para mapear as informações sobre esse ambiente. A intencionalidade dos atores, suas dependências, seus objetivos, tarefas e recursos são exemplos de conceitos que podem ser modelados com essa tecnologia. Além de oferecer vários construtores, há dois tipos de modelos para descrever esses requisitos: modelo de dependência estratégica (MDE) e modelo de razão estratégica (MRE).

O iStar descreve informações aderentes aos atributos de transparência, conforme relatado no trabalho de Leite e Cappelli (2008), ou seja, o *framework* possui características que permitem identificar atributos de transparência. A intencionalidade contribui para atingir transparência, pois através dela é possível obter *rastreabilidade* e *verificabilidade*, consequentemente contribui para que o grau de transparência *auditabilidade* seja alcançado, conforme mostra a Figura 1.

2.3 O *Framework* iStar

O iStar é um *framework* de modelagem de requisitos orientado a objetivos que permite representar o relacionamento de dependência entre os *atores*. Um ator é uma entidade que realiza tarefas de acordo com seus conhecimentos para atingir sua meta (LEITE et al., 2007). Quando o ator desempenha suas atividades ele pode empregar recursos para que essas sejam finalizadas e consequentemente o objetivo seja alcançado. As Tabelas 2 e 3 exibem os elementos intencionais e os conectores do iStar.

Tabela 2: Elementos intencionais do iStar.
Fonte: iStar Guides[4].

Elemento intencional	**Descrição**
Goal	Representa as intenções, desejos e objetivos de um ator.
Softgoal	Semelhante ao *Goa*l, porém, os critérios para a satisfação do objetivo dependem do ponto de vista do ator.
Task	Ações realizadas pelo ator para atingir seu objetivo.
Resource	Material/Informação utilizado durante a execução de uma tarefa.
Actor	Entidade que realiza tarefas de acordo com seus conhecimentos para atingir sua meta.
Belief	Conceito sobre o mundo considerado como verdade pelo ator.
Agent	Ator com manifestações físicas concretas.
Role	Caracterização abstrata do comportamento de um ator social dentro de um contexto especializado ou domínio de esforço.
Position	A *position* cobre um *role* e um *role* ocupa uma *position*.

Conforme descrito na Tabela 2, o ator é uma entidade que age de acordo com seus conhecimentos na execução de suas tarefas para alcançar um objetivo. Ele desempenha tarefas, consome recursos, colabora e recebe colaboração de outros atores, inseridos em seu ambiente, para alcançar seus objetivos, suas metas. Um agente é uma entidade especializada de um ator que possui manifestações físicas

[4] http://istar.rwth-aachen.de/tiki-index.php?page=i*+Guides&structure=i*+Wiki+Home

concretas, semelhantes às dos humanos. Ele ocupa posições e desempenha papéis, adicionalmente uma posição é um conjunto de papéis desempenhados por um agente.

Tabela 3: Conectores do iStar.
Fonte: iStar Guides[5].

	Conector	Descrição
	Means-end	Indica a relação entre algumas/várias tarefa e uma meta.
	Decomposition	A tarefa pode ser fatorada em: *Goal, Softgoal, Resource e Task.*
Associação entre atores	*Is part of*	Associação entre atores indicando a relação todo-parte.
	ISA	Relacionamento que indica generalização entre atores.
	Plays	Indica que um agente desempenha um papel.
	Occupies	Indica que um agente ocupa uma posição.
	INS	Indica que um agente é uma instância de outro.
***Links* de contribuição**	*Make*	Contribuição suficiente para que o *softgoal* seja alcançado.
	Break	*Contribui de modo que o softgoal não é alcançado.*
	Some+	Contribui positivamente com alguma força desconhecida.
	Some-	Contribui negativamente com alguma força desconhecida.
	Help	Contribui parcialmente para que o softgoal seja alcançado.
	Unknown	Contribui para um softgoal, porém não se sabe se positiva ou negativa.
	Hurt	Contribui parcialmente para que o *softgoal* não seja alcançado.
	Or	O elemento é satisfeito se algum de seus filhos é satisfeito.
	And	O elemento é satisfeito se todos os filhos são satisfeitos.

Uma tarefa, *Task*, consiste em um conjunto de ações que devem ser executadas por um ou mais atores para que um dado objetivo seja alcançado. Ela pode ser decomposta em outra sub tarefa, em uma meta, meta flexível, ou até mesmo em um recurso. Uma meta pode ser concreta, (*Goal)*, ou flexível, denominada de (S*oftgoal)*. Um *Goal* possui indicadores claros que mostram como deve ser satisfeito um dado objetivo. Porém um *softgoal* não possui claramente os critérios que devem ser observados para alcançá-lo.

[5] http://istar.rwth-aachen.de/tiki-index.php?page=i*+Guides&structure=i*+Wiki+Home

A modelagem dos recursos, (*resource*), utilizados pelas tarefas para alcançar o objetivo garante a operabilidade do ator na execução dessas tarefas. Um *resource* pode ser um material/informação que é consumido durante a execução de uma dada tarefa. Completando a lista dos construtores intencionais do iStar, o elemento crença, *Belief*, representa a perspectiva do ator sobre o mundo, considerada como verdade sob seu ponto de vista. Portanto uma crença pode influenciar diretamente os desejos e intenções do mesmo, influenciando indiretamente a meta a ser alcançada (YU et al., 2001).

A Figura 2 mostra a representação gráfica dos elementos e conectores citados nas Tabelas 2 e 3. Por exemplo, um *actor* é representado graficamente por um círculo, onde é indicado o seu nome. *Goal* é simbolizado por um retângulo de cantos aredondados. A geometria dos elementos faz sua distinção e mantém seu significado, porém sua cor pode variar de acordo com a ferramenta de modelagem iStar[6].

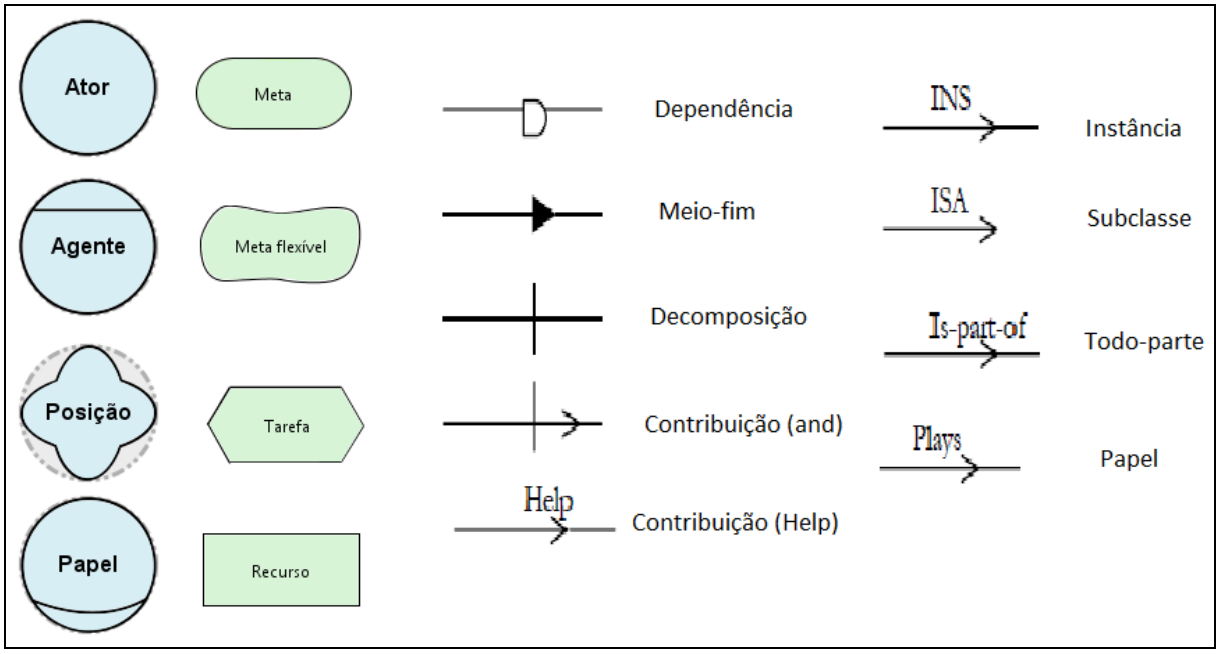

Figura 2: Representação gráfica dos elementos e conectores do iStar.
Fonte: Adaptado de Yu et al., 2001.

[6] http://istar.rwth-aachen.de/tiki-index.php?page=i*%20Tools

O diagrama de dependência estratégica é utilizado para representar os relacionamentos de dependência entre atores do ambiente organizacional. Ele é formado por um conjunto de nós e arcos, onde os arcos relacionam a dependência entre os atores e os nós representam esses atores. Entre o relacionamento de dependência podem ocorrer quatro tipos de dependência: dependência de meta, tarefa, recurso e meta flexível (LEITE et al., 2009).

A Figura 3 apresenta um exemplo desse diagrama onde a dependência entre os atores de um sistema de comércio eletrônico baseado em *buyer driven* foi modelada (YU et al., 2001). Existe uma dependência de *Goal* entre o cliente e o vendedor, ou seja, ele deseja encontrar um serviço com preço baixo, nesse sentido, depende do vendedor para que esse objetivo seja alcançado. O cliente é chamado de *depender*, o *Goal dependum* e o vendedor de *dependee*.

A Figura 3 mostra também a dependência de tarefa, onde o revendedor depende do pagamento da compra realizado pelo cliente, a dependência de recurso entre o fornecedor e o revendedor que depende de acordar preços com os fornecedores e por fim a dependência de softgoal, onde o fornecedor depende do aumento do número de clientes do revendedor.

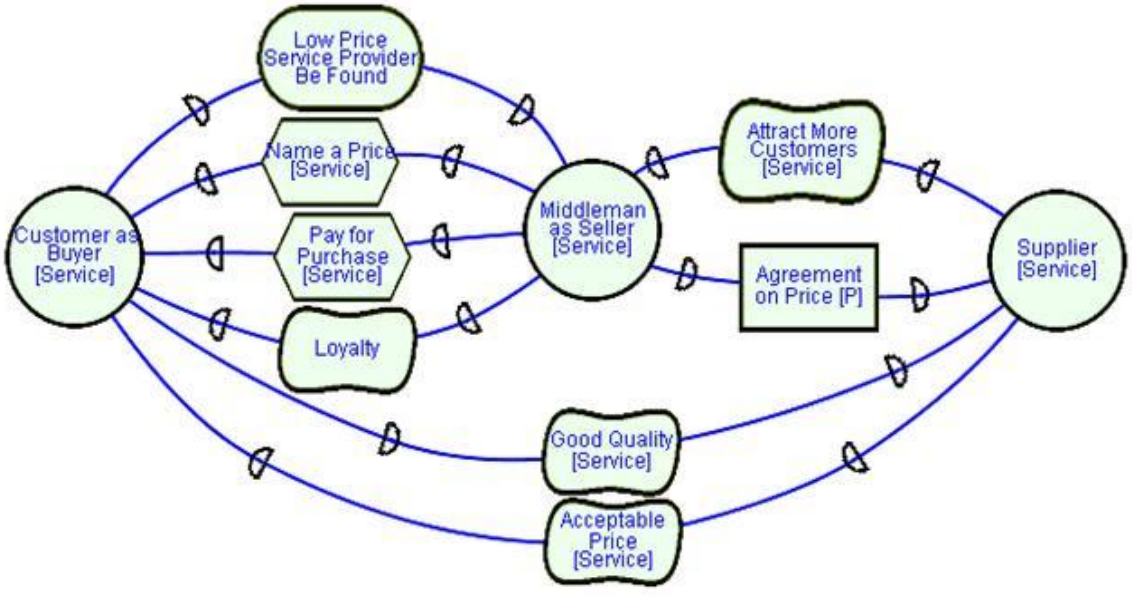

Figura 3: Diagrama de dependências do sistema buyer-driven e-commerce.
Fonte: Yu et al., 2001.

Além de representar as dependências entre os envolvidos no ambiente organizacional, o *Framework* iStar permite modelar racionalidade de atores através

do diagrama de razões estratégicas, SR (*Strategic Rationale*). Esse diagrama é formado pelos elementos intencionais, exibidos na Tabela 2, associados aos conectores de *means-end, decomposition e contributions.* O grafo formado pelos elementos intencionais permite representar a intencionalidade dos atores (LEITE et al., 2009).

A Figura 4 mostra o diagrama de razões estratégicas do sistema *buyer-driven e-commerce* (YU et al., 2001), onde estão representados dois atores com suas respectivas intencionalidades: *Customer* e o *Middleman*. Nesse modelo não foram apresentadas as intencionalidades do ator *Supplier*, apenas o relacionamento de dependência com os demais atores.

O principal objetivo do cliente é que um produto ou serviço seja adquirido. Para alcançar esse objetivo, ele desempenha a tarefa *Purchase By Naming My Own Price [Service]*, onde o cliente faz a oferta de preço pelo serviço e o vendedor decide se fecha a venda ou não. Essa tarefa está conectada ao *Goal* pela aresta *means-end.* Essa tarefa sofreu decomposição através de uma subtarefa e de um *Goal* usando o conector *decomposition: Name a Price [Service]* e *Low Price Service Provider Be Found.* Essa decomposição indica que o cliente faz a proposta de preço e que ele deseja que serviços de baixo preço sejam encontrados.

Na fronteira do cliente é possível observar ainda a presença de dois *Softgoals: Low Price* e *Flexibility.* Essa duas metas flexíveis indicam que o cliente deseja encontrar preços baixos e flexibilidade. Além disso, o conector do tipo *contribution* entre a subtarefa *Name a Price [Serviçe]* e a meta flexível *Low Price* indica que essa subtarefa vai ajudar (*Help*) a atingir a meta flexível de serviços com preços baixos. Por outro lado o *Goal Low Price Service Provider Be Found* dificulta a flexibilidade de escolha de serviços, pois as preferências à cerca de horários e datas, escolha da companhia, entre outras, não poderiam ser conciliadas.

O revendedor, *middleman,* tem como objetivo intermediar a oferta de serviços do fornecedor, conforme representado com o *Goal Be Middleman*. Este objetivo é alcançado pela execução da tarefa *Sell in Buyer driven style,* que por sua vez foi decomposta em outras três subtarefas: *Accept Purchase Request with Price, Send Modified Request to Supplier* e *Get Price Agreement From Supplier.*

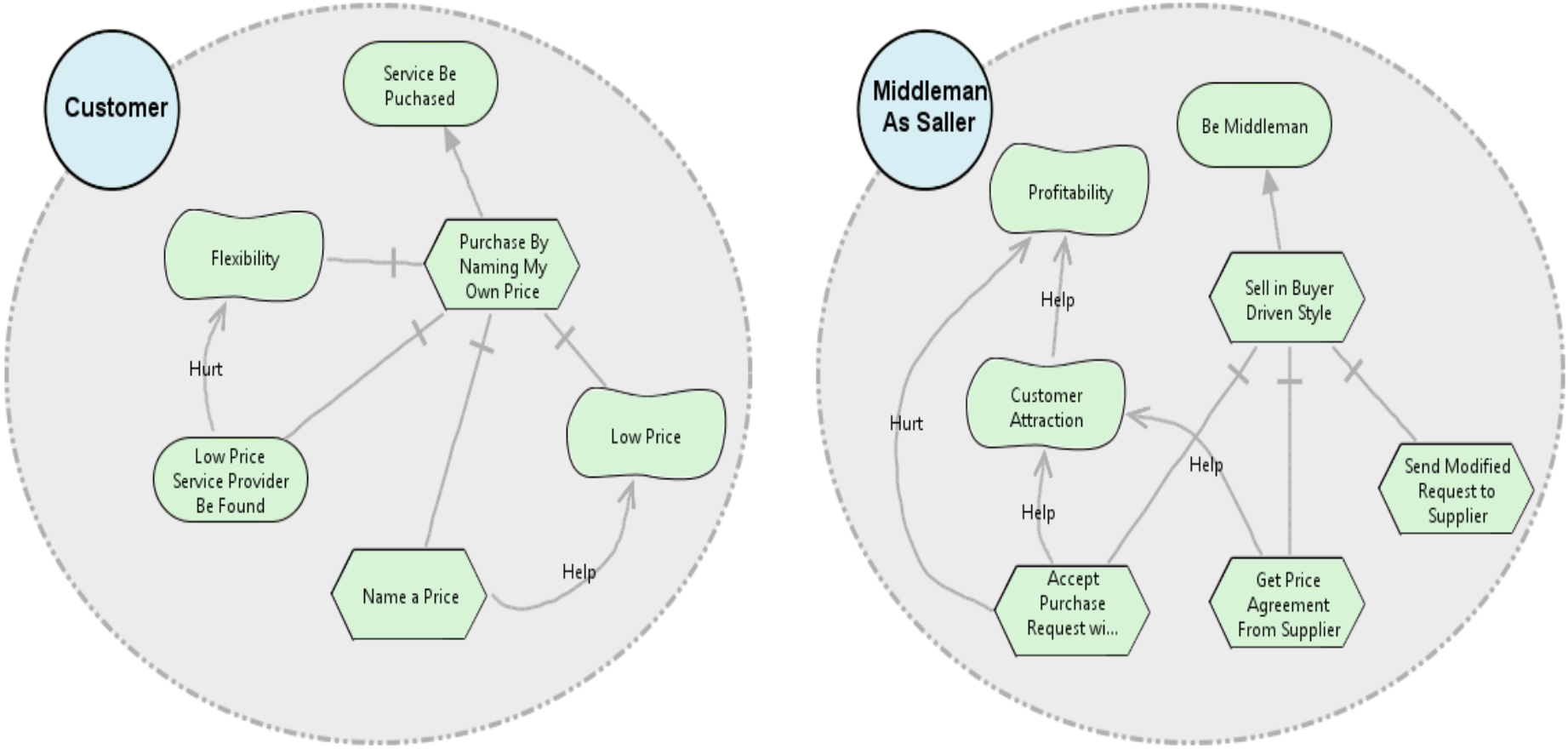

Figura 4: Diagrama de razões estratégicas do sistema buyer-driven e-commerce.

Fonte: Adaptado de Yu et al., 2001.

Para atuar como intermediário entre o comprador e o fornecedor, o revendedor deve aceitar pedido de compra com preço, enviar ao fornecedor o pedido modificado e negociar o preço do serviço com o fornecedor. Além disso, ele deseja obter rentabilidade e atrair novos clientes, conforme modelado com o elemento *softgoal.* As subtarefas *Accept Purchase Request with Price* e *Get Price Agreement From Supplier* ajudam a atrair novos clientes, porque quando o revendedor aceita o pedido com o preço do cliente e negocia esse preço com o fornecedor ele tende a fechar o negócio com o cliente. Porém, a subtarefa *Accept Purchase Request with Price* não contribui para que o objetivo rentabilidade seja alcançado, pois o preço ofertado pelo cliente naturalmente tende a ser menor.

O *Framework* iStar captura a intencionalidade dos atores através das metas e metas flexíveis, representa o conceito de tarefa, subtarefas, recursos e crenças, além de fornecer um mecanismo para especialização de atores. A dependência estratégica entre os atores são modeladas no modelo de dependência estratégica (LEITE e CAPPELLI, 2008).

2.4 Sistemas Especialistas

O Conhecimento é definido como o entendimento teórico ou prático sobre um determinado assunto. O indivíduo que detém profundo conhecimento sobre uma área específica é considerado especialista nesse domínio. Os sistemas especialistas foram desenvolvidos para simular o comportamento desses especialistas na tomada de decisão para solução de problemas em domínios específicos. (NEGNEVITSKY, 2004).

Os sistemas especialistas baseados em regras representam o conhecimento do especialista humano através do sistema de produção. O modelo de sistema de produção é baseado no princípio que o modelo de raciocínio humano utiliza regras de produção para resolver problemas (NEWELL e SIMON, 1972). O raciocínio ocorre através da aplicação das regras de produção aos dados armazenados na memória humana, obtendo assim as conclusões sobre o problema em questão.

Uma regra de produção pode representar uma relação, recomendação, diretriz, estratégia ou heurística. Ela é formada por duas partes: antecedente e consequente. Na parte antecedente contém a condição necessária para que a regra

seja executada, ou seja, se os fatos descritos nessa parte forem verdadeiros, então ocorre a execução da regra. Na parte consequente são especificadas as ações executadas quando forem satisfeitas as premissas especificadas na parte antecedente (NEGNEVITSKY, 2004).

Para exemplificar, considere a seguinte regra: ***se*** *o sinal está verde* ***então*** *prossiga*. A condição necessária para que haja o deslocamento do objeto é que o sinal esteja verde. Portanto uma regra assume a seguinte forma:

IF
antecedente$_1$ **OR**|**AND**|**NOT** antecedente$_2$, ..., **OR**|**AND**|**NOT** antecedente$_n$
THEN
consequente$_1$, consequente$_2$, ..., consequente$_n$

A parte antecedente pode combinar os conectivos lógicos *and, or e not* para expressar a relação entre as premissas. Na parte consequente são encadeadas as ações realizadas quando a regra for disparada. Após a execução dessas ações, conclusões são obtidas de acordo com o conhecimento representado pela referida regra.

Segundo Luger (2004) existem algumas diretrizes que indicam se o problema é apropriado para ser resolvido com sistemas especialistas, são elas:

- Relação custo/benefício positiva - a economia gerada compensa o investimento no desenvolvimento e manutenção do sistema especialista.
- Falta de perícia humana em situações necessárias - por exemplo, na atividade mineradora, onde há a necessidade da visita de geólogos e engenheiros. Com um sistema especialista custos podem ser reduzidos.
- O problema pode ser solucinado com raciocínio simbólico - quando a solução não necessitar de capacidade perceptivas.
- O domínio da aplicação é bem estruturado – O conhecimento é formal e específico.
- Existem especialistas humanos que tenham o interesse em compartilhar seus conhecimentos, pois é através do conhecimento deles que a base de conhecimento é formada.

- O escopo do problema é adequado – Não há como capturar todo o conhecimento de um médico, porém é possível desenvolver um sistema especialista para diagnóstico de uma doença.

A Figura 5 apresenta a estrutura de um sistema especialista baseado em regras de produção. Esse modelo de representação e inferência de conhecimento é formado por cinco componentes:

- *knowledge base* – O conhecimento extraído do especialista humano é representado no sistema especialista através de regras de produção formando assim sua base de conhecimento.
- *database* – A base de dados do sistema é formada pelos dados sobre o problema em questão, os fatos.
- *inference engine* – O mecanismo de inferência é responsável pela aplicação das regras de produção à base de fatos para obter a solução para o problema proposto.
- *explanation facilities* – Um sistema especialista deve demonstrar como uma solução foi alcançada ou por que um fato específico é necessário para resolver o problema. Este módulo é responsável por revelar essas informações aos usuários do sistema.
- *user interface* – A interface de usuário é o meio de comunicação entre o usuário e o sistema especialista, possibilitando que o mesmo adicione informações à base de dados, formule regras e execute consultas.

Além dos cinco módulos discutidos anteriormente, a estrutura do sistema contém ainda a interface de desenvolvimento para que os desenvolvedores possam editar a base de conhecimento. Por fim, o motor de inferência pode interagir com componentes externos ao sistema, por exemplo, quando há a necessidade de extração de dados armazenados em bases externas ou processamento realizado por programas escritos em linguagens de programação convencionais.

O mecanismo de inferência de um sistema especialista, representado pelo componente *Inference engine* na Figura 5*,* aplica regras de produção aos fatos para inferir conclusões sobre o problema em questão. Ele compara cada condição da parte antecedente das regras na base de conhecimento com os fatos da base de dados, quando há correspondência é disparada a respectiva ação. Ou seja, se há

fatos na base de dados que satisfazem às premissas das regras, então as ações descritas na parte consequente são executadas. Nesse sentido, duas estratégias de inferência são utilizadas: *Forward chaining ou backward chaining.*

O encadeamento para frente (*forward chaining)* é uma técnica de inferência baseada em dados. O processo de encadeamento inicia com os dados conhecidos e prossegue comparando os fatos da base com as regras. O encadeamento pára quando não há mais fatos que correspondam às premissas das regras.

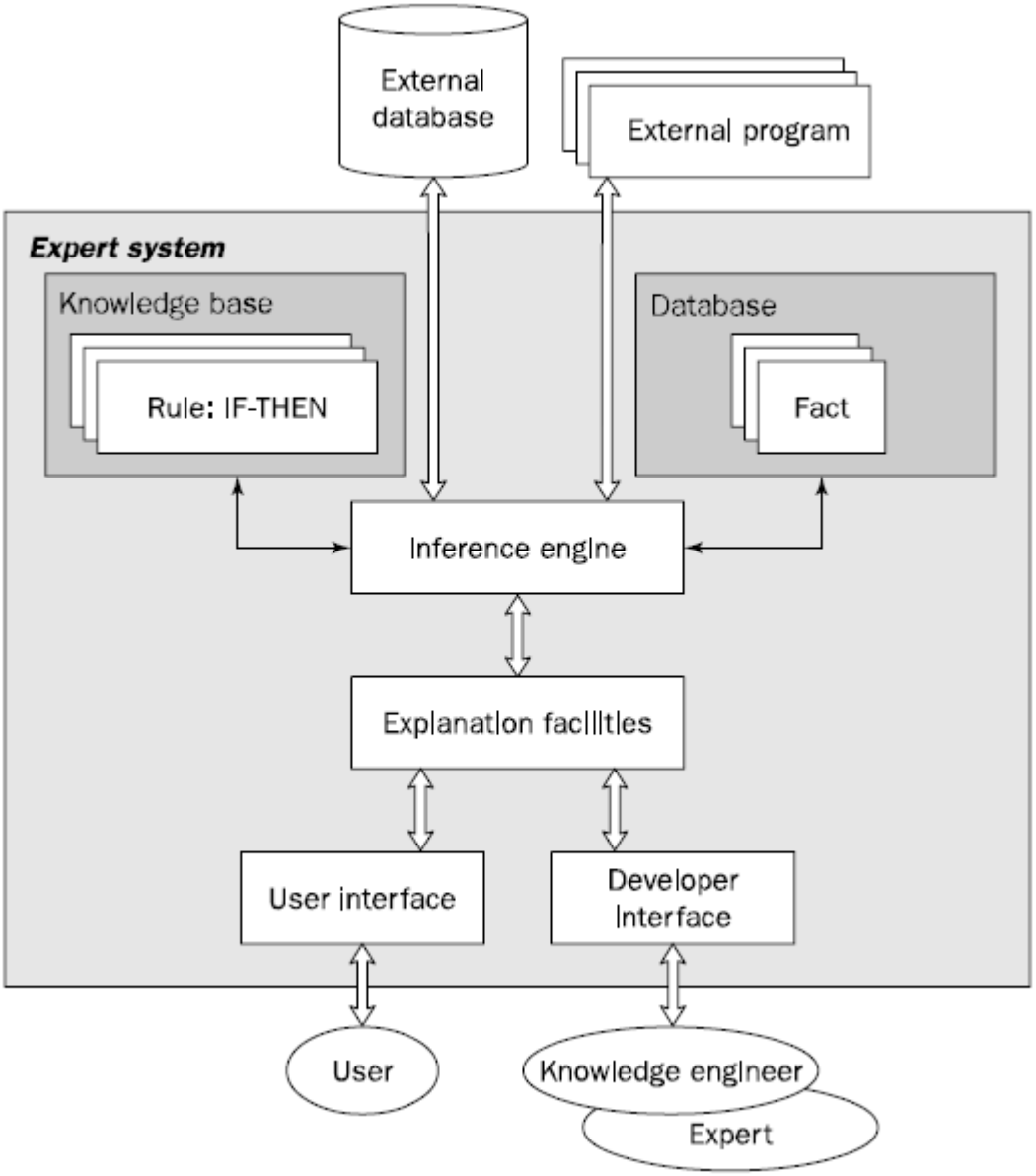

Figura 5: Estrutura de um sistema especialista baseado em regras.
Fonte: Negnevitsky, 2004.

A Figura 6 exemplifica o processo de inferência *forward chaining.* Inicialmente a base de dados contém cinco fatos, quando o motor de inferência analisa a regra A→X, é verificado que na base de fatos existe o fato A, então o fato X é adicionado na base. Dessa forma o processo vai analisando a correspondência de fatos na base com as condições das regras e executando as ações especificadas até que não haja mais correspondencia entre fatos e premissas.

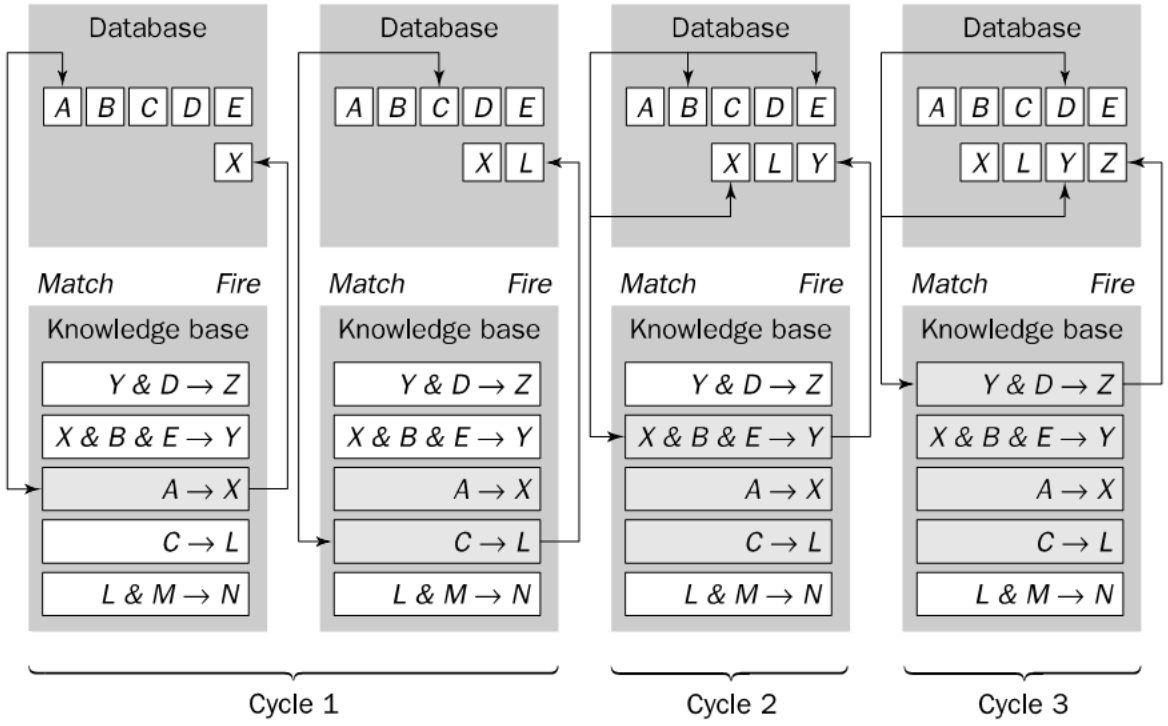

Figura 6: Encadeamento para frente.

Fonte: Negnevitsky, 2004.

O encadeamento para trás (*backward chaining)* é uma técnica de inferência baseada em objetivos. A partir de uma hipótese procura-se evidências para prová-la. A base de conhecimento é examinada para encontrar regras que solucionam o problema em questão. Se essa solução for encontrada em uma regra e sua respectiva parte antecedente corresponder aos fatos na base de dados, então ela é disparada e o objetivo é provado. Porém, se isso não ocorrer, o motor de inferência empilha essa regra e estabelece um sub-objetivo para provar. Esse processo se repete até que esse sub-objetivo seja provado, então o objetivo principal o será.

A Figura 7 apresenta um exemplo dessa técnica de inferência. O objetivo é provar que Z é solução para o problema, então é examinada a base de conhecimento para encontrar uma regra que tenha Z na parte consequente.

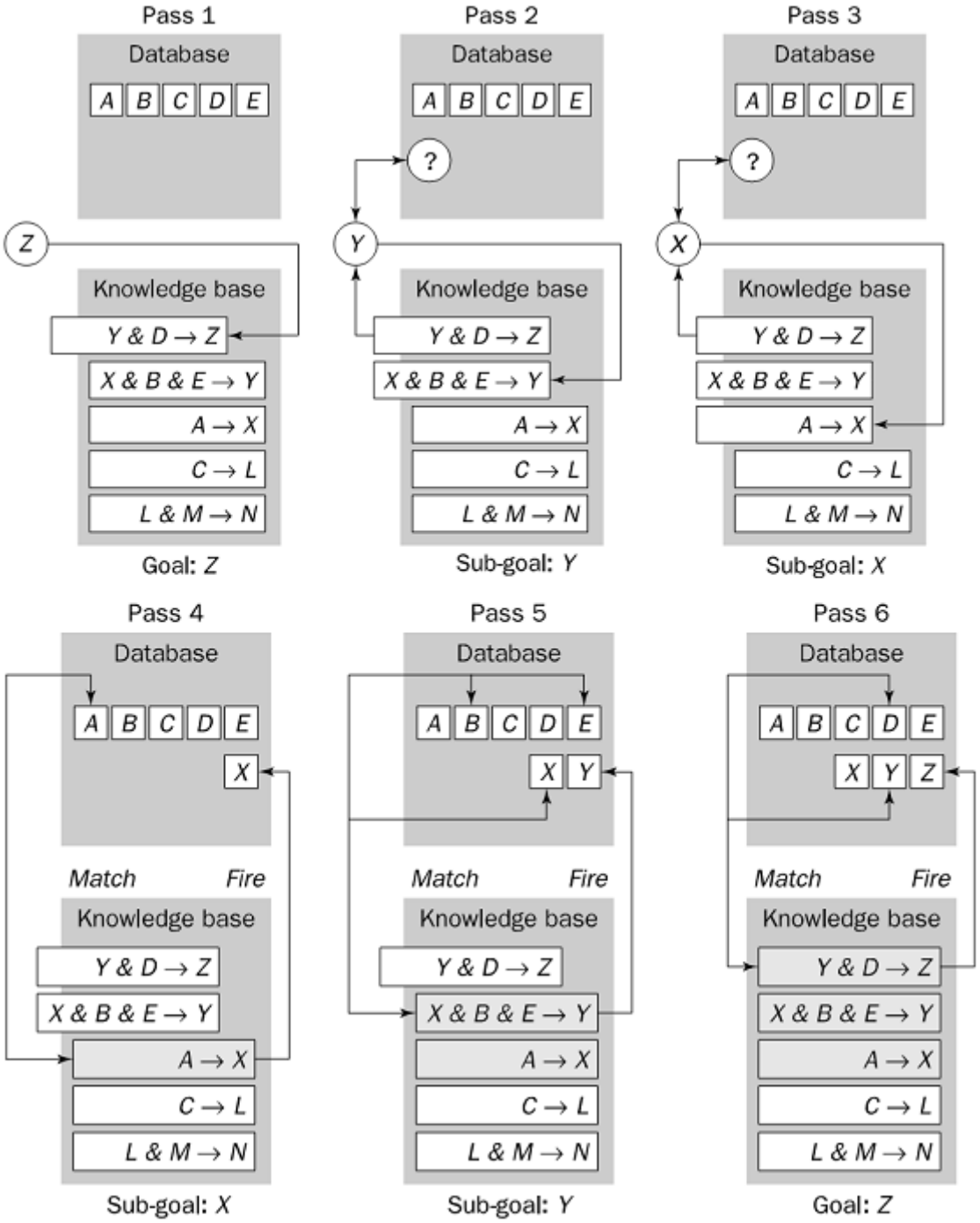

Figura 7: Encadeamento para trás.

Fonte: Negnevitsky, 2004.

A primeira regra da Figura 7 tem o fato Z como ação, porém Y e D não fazem parte da base de fatos. Nesse momento a regra Y & D→Z é empilhada. O próximo passo é tentar provar Y, depois provar D é então finalmente provar Z. A segunda regra prova Y, porém X não está na base, então empilha-se 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.

O encadeamento para frente (*forward chaining)* é indicado para sistemas de análise e interpretação, enquanto o encadeamento para trás (*backward chaining)* é indicado para sistemas de diagnósticos, embora ambas estratégias de raciocínios possam ser implementadas no mesmo mecanismo de inferência (NEGNEVITSKY, 2004).

2.5 CLIPS - C *Language Integrated Production System*

CLIPS é uma ferramenta para construção de sistemas especialistas originalmente desenvolvida na década de oitenta no Software Technology Branch (STB), NASA/Lyndon B. Johnson Space Center (GIARRATANO, 1993).

CLIPS foi desenvolvido com o propósito de facilitar o desenvolvimento de sistemas especialistas. O conhecimento é representado através de regras de produção e a base de dados é composta de fatos. Além disso, é possível utilizar funções e orientação a objetos. O mecanismo de inferência é baseado na técnica de encadeamento progressivo (*forward chaining*) (GIARRATANO, 1993).

A Figura 8 exemplifica a criação de uma base de dados juntamente com sua base de regras para um sistema de controle de tráfego com o objetivo de mostrar de forma didática o funcionamento dessa ferramenta. Nas linhas 1 à 4 são declaradas a estrutura de dados *deftemplate sinal* e *veículo* para armazenar as informações do sinal de trânsito e do veículo respectivamente. Nas linhas 6 e 7 é adicionado à base de dados o fato *sinal*, indicando que o mesmo está verde.

A regra para controlar o fluxo de veículo foi declarada nas linhas 9 à 13. Ela verifica se na base de fatos há o fato *sinal* com *status* verde, se afirmativo então o fato veículo com *status* prosseguir será adicionado aos fatos.

```
1  (deftemplate sinal
2       (slot status  (type STRING) )      )
3  (deftemplate veiculo
4       (slot status  (type STRING) )      )
5
6  (deffacts transito "Informações sobre o trânsito"
7      (sinal(status "verde") )            )
8
9  (defrule trafegar
10     (sinal (status "verde") )
11     =>
12     (assert (veiculo (status "prosseguir")) )
13 )
```

Figura 8: Base de dados e conhecimento em CLIPS.

A Figura 9 mostra o resultado obtido pelo motor inferência CLIPS. Quando a base de dados juntamente com a base de conhecimento é carregada, a base de fatos contém somente um fato, *(sinal(status "verde"))*, quando essas informações são submetidas ao motor de inferência pela execução do comando *(run),* a regra *trafegar* é examinada e sua premissa é satisfeita, nesse momento a regra é disparada e o novo fato é adicionado à base de dados. Observe na linha 30 o fato adicionado pela execução da regra *trafegar*.

```
15 CLIPS> (load "C:/Users/Joás/Google Drive/my files/Mestrado/Minha dissertação/fatos.clp")
16 Defining deftemplate: sinal
17 Defining deftemplate: veiculo
18 Defining deffacts: transito
19 Defining defrule: trafegar +j+j
20 TRUE
21 CLIPS> (reset)
22 CLIPS> (facts)
23 f-0     (initial-fact)
24 f-1     (sinal (status "verde"))
25 For a total of 2 facts.
26 CLIPS> (run)
27 CLIPS> (facts)
28 f-0     (initial-fact)
29 f-1     (sinal (status "verde"))
30 f-2     (veiculo (status "prosseguir"))
31 For a total of 3 facts.
32 CLIPS>
```

Figura 9: Resultado da inferência obtida com o sistema de controle de tráfego.

3 MÉTODO DE DESENVOLVIMENTO DO TRABALHO

Esse capítulo apresenta o método utilizado para verificar a presença de requisitos de transparência em especificações de requisitos de *software*. A Figura 10 apresenta o fluxo das atividades desenvolvidas para alcançar esse objetivo. Cada atividade está associada ao número da seção onde a mesma é apresentada.

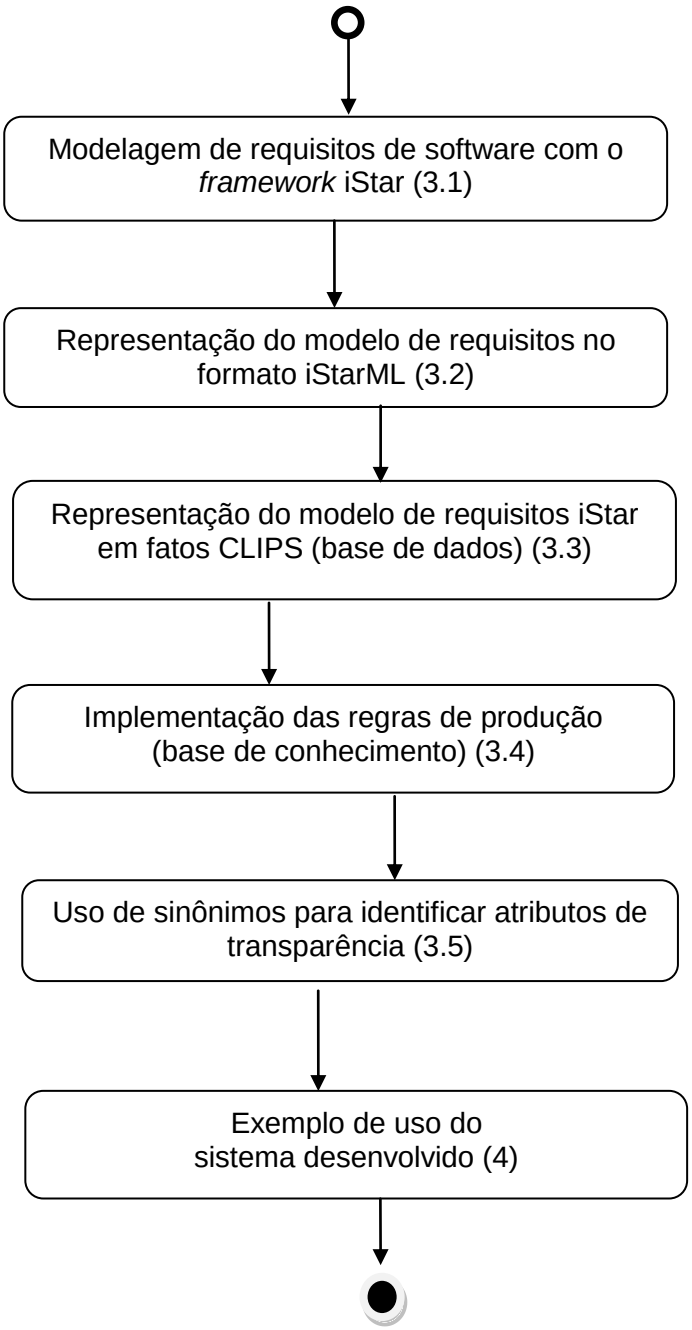

Figura 10: Atividades realizadas no método de desenvolvimento do trabalho.

Inicialmente os requisitos do sistema são especificados utilizando o *Framework* iStar. A utilização desse *Framework* tem como objetivo usar suas características para identificar atributos de transparência, conforme apresentado na seção 3.1, (LEITE e CAPPELLI, 2008).

A partir da utilização do *Framework* iStar para especificar os requisitos do sistema, buscou-se uma representação desses requisitos independente de plataforma, nesse sentido foi utilizado o formato iStarML, um formato de intercâmbio de modelos iStar entre ferramentas de modelagem. A utilização desse formato provê uma interface de comunicação entre as ferramentas de modelagem de requisitos iStar (CARES et al., 2008). A seção 3.2 apresenta a abordagem para representar um modelo de requisitos nesse formato.

A base de dados do sistema de verificação é formada pela especificação de requisitos representada no formato de fatos do sistema CLIPS (GIARRATANO, 1993). Essa atividade permite que os dados sobre o modelo de requisitos sejam submetidos ao motor de inferência do CLIPS. Na seção 3.3 é discutida a abordagem utilizada para formar a base de dados a partir da especificação de requisitos representada com o *Framework* iStar.

A próxima atividade desenvolvida foi implementar as regras de produção que verificam a aderência da especificação de requisitos aos atributos de transparência. Para implementá-las foram utilizadas as características providas pelo *Framework* iStar que indicam os requisitos de transparência, conforme apresentado na seção 3.4, (LEITE e CAPPELLI, 2008).

A identificação de sinônimos dos atributos de transparência consiste em reunir termos que possam sugerir a existência de atributos de transparência. Assim é possível verificar se no modelo elas foram mencionadas. A partir da consulta ao WordNet foram identificados os termos que são indícios desses atributos. O WordNet é uma base de dados de palavras da língua Inglesa que reúne substantivos, verbos, adjetivos e advérbios em conjuntos de sinônimos cognitivos (WORDNET, 2012).

Os atributos de transparência e seus termos relacionados foram representados em fatos, conforme a sintaxe da linguagem CLIPS. Esse passo é necessário, pois a regra de produção que identifica se a especificação de requisitos menciona os atributos de transparência ou seus termos relacionados atuará sobre

esses fatos e sobre os fatos que representam o modelo de requisitos. Na seção 3.5 é apresentada a estratégia para construir a base de sinônimos dos atributos de transparência.

Por fim foi utilizado um exemplo da literatura para avaliar a base de conhecimento desenvolvida.

3.1 Modelagem de Requisitos de Software com o *Framework* iStar

Os requisitos do sistema são especificados com o *Framework* iStar. Essa abordagem permite utilizar informações do iStar para verificar a presença de requisitos de transparência, pois esse *Framework* possui características que indicam atributos de transparência. Intencionalidade, alternativas de operacionalização, dependência estratégica e decomposição de tarefas são exemplos dessas características (LEITE e CAPPELLI, 2008).

A Tabela 4 apresenta os atributos de transparência relacionados com as informações que permitem identificá-los. Essas características são usadas como conhecimento na implementação das regras de produção para indicar a presença de atributos de transparência em especificações de requisitos representadas com *Framework* iStar.

A coluna *Relação* da Tabela 4 expressa a correspondência entre as características do iStar identificadas pelos algarismos de 1 a 9 na coluna *ID*, com os atributos de transparência. *Rastreabilidade*, por exemplo, é relacionada com as características 1, 5.2 e 9 do iStar.

A coluna *Graus de transparência* agrupa os requisitos de acordo com sua contribuição para a transparência. Por exemplo, os requisitos *verificabilidade*, *controlabilidade*, *rastreabilidade*, *validação* e *intencionalidade detalhada* colaboram para que o nível de transparência *auditabilidade* seja alcançado.

Segundo Leite e Cappelli (2008) a intencionalidade modelada com o iStar indica *rastreabilidade* e *verificabilidade* das ações realizadas pelos atores do ambiente organizacional. O detalhamento dessas intenções indica *compositividade* e *divisibilidade* das ações realizadas pelo ator para alcançar seus objetivos. A modelagem de *softgoals* indica *clareza*, *completeza* e *acurácia*. Por fim as *alternativas de operacionalização* sinalizam os atributos *extensibilidade*, *integridade*

e *validade*. As características identificadas com os algarismos 5 à 9 foram mencionadas no trabalho de Oliveira et al. (2007).

Tabela 4: Características que indicam requisitos de transparência.
Fonte: Leite e Capelli, 2008 e Oliveira et al., 2007.

	Atributos de transparência	Relação	ID	Característica iStar
Acessibilidade	Portabilidade		1	Intencionalidade (*actor* e *goal*)
	Disponibilidade	6	2	*Softgoal*
	Publicidade		3	Intencionalidade detalhada (decomposição de tarefas)
Usabilidade	Uniformidade		4	Alternativas de operacionalização (conectores *means-end*)
	Simplicidade			
	Operabilidade	5.1	5	Dependência estratégica
	Intuitividade		5.1	Dependência de recurso
	Desempenho	8	5.2	Dependência de meta
	Adaptabilidade		5.3	Dependência de meta flexível
	Amigabilidade		6	Classificação de atores (Associação entre atores INS e ISA)
Informatividade	Clareza	2, 5.2, 9	7	Estrutura organizacional (posições)
	Completeza	2	8	Responsabilidades (papeis)
	Corretude		9	Contribuição positiva/negativas (*link contribution*)
	Atualidade			
	Comparabilidade			
	Consistência			
	Integridade	4		
	Acurácia	2		
Entendibilidade	Concisão			
	Compositividade	3, 5.3		
	Divisibilidade	3, 8		
	Detalhamento	4		
	Dependência	5, 9		
Auditabilidade	Validável	4		
	Controlabilidade			
	Verificabilidade	1, 5.2, 7		
	Rastreabilidade	1, 5.2, 9		
	Explicável			

Conforme identificado por Leite e Cappelli (2008) e Oliveira et al. (2007), o *Framework* iStar possui várias características que permitem sinalizar a presença dos atributos de transparência em especificações de requisitos modeladas com esse *framework*. Porém, a partir da análise da Tabela 4 é possível observar vários atributos que não possuem relação com os conceitos do iStar. Por exemplo, o atributo *portabilidade* não está relacionado a nenhuma característica, impossibilitando sua identificação a partir de uma especificação iStar. Além de utilizar os conceitos do *Framework* iStar para implementar a base de conhecimento, foram utilizados sinônimos dos atributos de transparência para indicar a presença de palavras na especificação de requisitos que indicam os atributos de transparência.

3.2 Representação do Modelo de Requisitos no Formato iStarML

O iStarML é um formato baseado na tecnologia XML (*Extensible Markup Language*) proposto para oferecer intercâmbio de modelos de requisitos entre as ferramentas de modelagem iStar[7]. A escolha desse formato para representar os requisitos agrega os seguintes benefícios (CARES et al., 2008):

- Um formato para o intercâmbio de especificações de requisistos entre as diversas ferramentas de modelagem iStar.
- Estimula a adoção de um formato de intercâmbio de modelos iStar.
- Desenvolvimento de algoritmos de análise de especificações de requisitos independentes de representações gráficas.
- Evolução das ferramentas de modelagem iStar com novas componentes de análise baseadas nos conceitos iStar.
- Representar as diferenças e semelhanças entre as variantes existentes do iStar.
- Representar restrições sintáticas sobre os modelos iStar.

A Tabela 5 mostra as *tags* disponíveis para representar os elementos intencionais do *Framework* iStar. O elemento *actor* do *Framework* iStar é representado pela tag <*actor/*> e para representar a classificação desses atores é utilizado o atributo *type.*

[7] http://istar.rwth-aachen.de/tiki-index.php?page=i*%20Tools

Tabela 5: Tags iStarML.

Fonte: Cares et al., 2008.

Elementos do iStar	Tags iStarML	Principais atributos ou subtags
Actor	<actor>	Atributo *type* especifica os tipos de atores.
Elementos intencionais	<ielement>	Atributo *type* especifica os tipos de elementos. Por exemplo o tipo *goal*.
Depedência	<dependency>	Pode conter duas subtags <dependee> e <depender>
Boundary	<boundary>	Especifica a fronteira de um ator.
Link de elemento intencional	<ielementLink>	Especifica os tipos de relações intencionais entre os elementos do iStar.
Link de de atores	<actorLink>	O atributo *type* especifica o tipo de associação entre os atores. Por exemplo: *is_part_of*.

A Figura 11 apresenta o modelo de razões estratégicas do cenário *adicionar foto* em um dispositivo móvel. O usuário deseja manipular mídias, e para manipular uma foto ele informa o local onde a mesma está, atribui seu nome e indica o álbum ao qual ela pertencerá. O sistema do dispositivo é o ator responsável pelo gerenciamento das mídias manipuladas pelo usuário. Para alcançar esse objetivo o sistema inicializa o aplicativo, disponibiliza a lista de álbuns, salva e armazena a fotos.

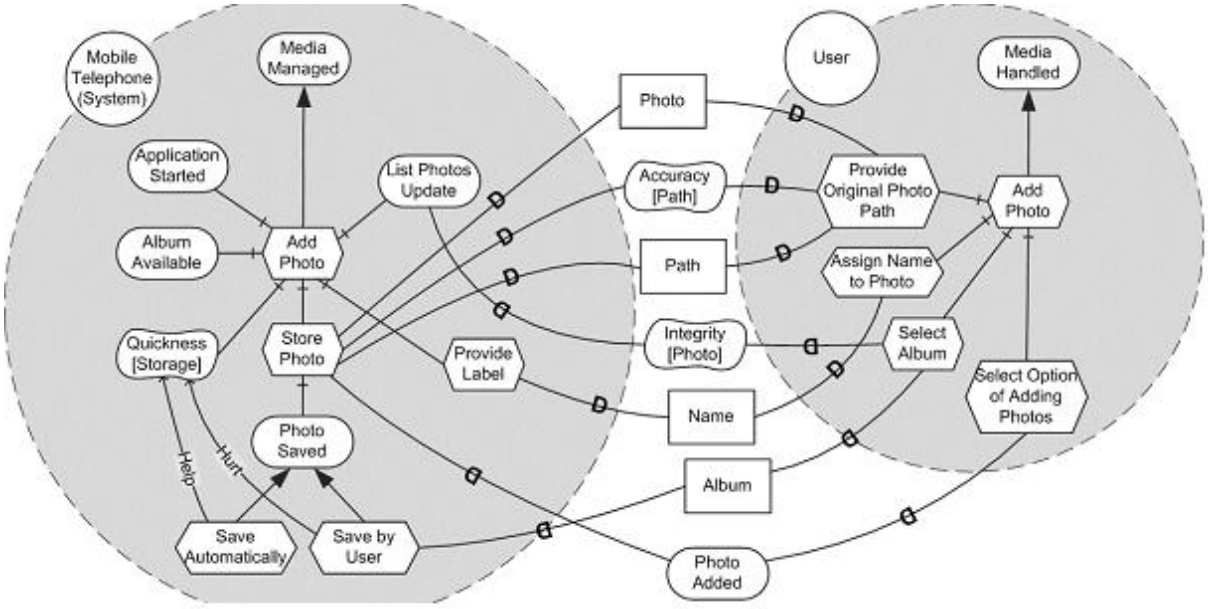

Figura 11: Modelo de razões estratégicas do cenário adicionar foto.

Fonte: Adaptado de Silva et al., 2011.

O modelo de razões estratégicas apresentado na Figura 11 foi representado no formato iStarML. A Figura 12 ilustra parcialmente o modelo de requisitos da

Figura 11, onde o diagrama está aninhado dentro da *tag* <*istarml/*> conforme as linhas 2 à 17, os elementos intencionais do iStar são aninhados pela *tags* <*actor/*> e <*boundary/*> indicando que cada ator no modelo tem uma fronteira, linhas 4 à 15, e dentro dessa há os elementos intencionais do framework iStar. O elemento *manipular mídia* foi representado pela *tag* <*ielement/*> para indicar que o mesmo é um elemento intencional.

```
1   <?xml version="1.0"?>
2   <istarml version="1.0">
3       <diagram name="Mobile Media">
4           <actor id="01" name="usuário">
5               <boundary>
6                   <ielement id ="02" type="goal"
7                       name="Manipular midia"/>
8               </boundary>
9           </actor>
10          <actor id="03" name="Telefone móvel">
11              <boundary>
12                  <ielement id ="04" type="goal"
13                      name="Gerenciar midia"/>
14              </boundary>
15          </actor>
16      </diagram>
17  </istarml>
```

Figura 12: Cenário adicionar foto representado parcialmente em iStarML.

Observa-se na Figura 12 que vários elementos do diagrama de razões estratégicas do cenário *adicionar foto* foram omitidos porque o arquivo XML original é muito grande para ser representado em uma única imagem sem prejuízo da legibilidade.

Esse formato de representação de especificações iStar foi escolhido para prover um sistema de verificação de requisitos independente de plataforma ou ferramenta de modelagem de requisitos iStar.

3.3 Representação do Modelo de Requisitos em Fatos Clips.

O modelo de requisitos representado com o iStarML deve ser transformado no formato de fatos da linguagem CLIPS para que o mecanismo de inferência do sistema CLIPS possa aplicar as regras de produção para extrair o conhecimento

implícito nos requisitos representados nesses fatos. Esse conhecimento são os atributos de transparência.

A Tabela 6, mostra a relação dos fatos criados para representar o modelo de requisitos em iStar. Todos os elementos do *Framework* iStar são representados pelo fato *element*. Os conectores que relacionam os elementos são representados pelo fato *elementLink*. Quando *element* representa os elementos *actor, agent, role e position,* o atributo *idActor* pode ser omitido, porque o atributo *idActor* representa a fronteira a que o elemento intencional pertence.

Tabela 6: Mapeamento entre os elementos iStar e fatos CLIPS.

Elementos do iStar	**Fatos CLIPS**
actor *agent* *role* *position* *goal* *softgoal* *task* *resource*	element((id “ ”) (name “ ”) (type “ ”) (idActor “ ”))
links	elementLink((id “ ”) (label “ ”) (type “ ”) (source “ ”) (target “ ”))

Para representar um fato em CLIPS é preciso definir um *template* para indicar quais atributos o fato armazenará. Na Figura 13 é exibido os templates *element e elementLink* que definem os atributos de *element* e *elementLink* respectivamente.

```
1  (deftemplate element
2       (slot id      (type STRING) )
3       (slot name    (type STRING) )
4       (slot type    (type STRING) )
5       (slot idActor (type STRING) )
6  )
7
8  (deftemplate elementLink
9       (slot id      (type STRING) )
10      (slot label   (type STRING) )
11      (slot type    (type STRING) )
12      (slot source  (type STRING) )
13      (slot target  (type STRING) )
14 )
15
```

Figura 13: Templates *element* e *elementLink*.

Nas linhas 1 à 6, da Figura 13, foi definido o *template element,* onde cada atributo é declarado juntamente com o seu tipo de dado. No modelo de razões

estratégicas do cenário *adicionar foto*, Figura 11, o elemento *manipular mídia* é representado com o seguinte fato:

(element (id "03") (name "Midia Managed") (type "goal") (idActor "01"))

A representação da especificação de requisitos em fatos CLIPS forma uma base de dados que permite que esses requisitos sejam submetidos ao motor de inferência do sistema CLIPS. Esse mecanismo de inferência aplica as regras de produção para identificar nessa especificação os atributos de transparência.

A partir de uma especificação de requisitos representado no formato iStarML, é possível obter a base de fatos do sistema CLIPS. A Tabela 7 mostra a correspondência entre as tags iStarML e os fatos *element* e *elementLink*.

Tabela 7: Correspondência entre o formato istarML e fatos em CLIPS.

Tag IstarML	**Fatos CLIPS**
<actor/>	*element((id “ ”) (name “ ”) (type “ ”))*
<ielement/>	*element((id “ ”) (name “ ”) (type “ ”) (idActor “ ”))*
<ielementLink/>	*elementLink((id “ ”) (label “ ”) (type “ ”) (souce “ ”) (target “ ”))*
<boundary/>	Mapeado para o atributo idActor do fato element (... (*idActor* “ ”))
<diagram/>	-
<istarml/>	-

Conforme exibe a Tabela 7, as *tags <actor/>* e *<ielement/>* são representadas pelo fato *element*, quando representa um elemento intencional o atributo *idActor* é informado. Essa estratégia permite relacionar os elementos intencionais do iStar com o respectivo ator, além disso representa a *tag <boundary/>.*

O relacionamento entre os elementos intencionais representado pela *tag <ielementLink/>* é mapeado no fato *elementLink.* As tags *<diagrams/>* e *<istarml/>* não são utilizadas pelas regras de produção, portanto não foram representadas na base de dados.

Na criação da base de fatos são utilizadas duas abordagens para transformar uma especificação de requisitos representada com o iStar para fatos CLIPS. A primeira tem como ponto de partida a especificação de requisitos em iStar que é transformada diretamente em fatos CLIPS. A segunda, inicia a partir da

especificação no formato iStarML que é transformada em fatos utilizando a informação da Tabela 7.

3.4 Representação do Conhecimento

O mecanismo de inferência do sistema CLIPS extrai o conhecimento implícito na base de fatos do sistema especialista através da aplicação das regras de produção. Quando houver correspondência entre as condições estabelecidas na parte antecedente da regra, ela será disparada e todas as ações estabelecidas na seção consequente serão executadas. A partir das características do *Framework* iStar foram desenvolvidas as regras de produção que verificam a presença ou ausência de requisitos de transparência em especificações de requisitos de *software* representadas com esse *Framework*.

Uma regra de produção pode ser dividida em duas partes: a seção *antecedente* e *consequente*. Na primeira estão as condições necessárias para que a regra seja disparada e na segunda parte, a ação a ser realizada quando todas as condições estabelecidas na parte antecedente são verdadeiras.

A partir da Tabela 4, foram criadas as regras de produção exibidas na Tabela 8. A regra *ruleIntentionality* identifica a intencionalidade representada na especificação de requisitos. Ela implementa o conceito de intencionalidade provido pelo iStar utilizando os construtores ator e meta. Para implementar as demais regras listadas na coluna *Regras de produção* da Tabela 8 foram utilizados os elementos descritos na coluna *Construtores do iStar* dessa mesma tabela.

Além da identificação de atributos de transparência, nesse trabalho foram desenvolvidas regras para apontar a falta desses atributos em especificações de requisitos representadas com o *Framework* iStar. As regras com o prefixo *ruleNot* foram projetadas para esse fim. Por exemplo, a regra *ruleNotIntentionality* indica atores que não possuem metas relacionadas em sua fronteira de atuação identificando a falta de intencionalidade na especificação de requisitos.

A partir da Tabela 8 é possível identificar os elementos, conectores e os conceitos do iStar que foram utilizados no desenvolvimento de cada regra.

Tabela 8: Regras de produção que formam a base de conhecimento.

Regras de produção	Características iStar	Construtores do iStar
ruleIntentionality	Intencionalidade	Ator e meta
ruleSoftgoal	Metas flexíveis	Meta flexível
ruleIntentionalityDetailed	Intencionalidade detalhada	Decomposição de tarefas
ruleAlternativeOfOperationalization	Alternativas de operacionalização	Conector *means-end*
ruleStrategicDependenceOfResource	Dependência de recurso	Ator e recurso
ruleStrategicDependenceOfGoal	Dependência de meta	Ator e meta
ruleStrategicDependenceOfSoftgoal	Dependência de meta flexível	Ator e meta flexível
ruleTaxonomyOfActors	Classificação de atores	Associação entre atores INS e ISA
ruleOrganizationalStructure	Estrutura organizacional	Posições
ruleResponsibility	Responsabilidades	Papéis
ruleContribution	Contribuição positiva/negativas	Conector *contribution*
ruleNotIntentionality	Intencionalidade	Ator e meta
ruleNotSofgoal	Metas flexíveis	Meta flexível
ruleNotDetailedIntention	Intencionalidade detalhada	Decomposição de tarefas
ruleNotAlternativeOfOperationalization	Alternativas de operacionalização	Conector *means-end*
ruleNotDependence	Dependência de recurso	Ator e recurso
	Dependência de meta	Ator e meta
	Dependência de meta flexível	Ator e meta flexível
ruleNotTaxonomyOfActors	Classificação de atores	Associação entre atores INS e ISA
ruleNotOrganizationalStructure	Estrutura organizacional	Posições
ruleNotResponsibility	Responsabilidades	Papeis
ruleNotContribution	Contribuição positiva/negativa	Conector *contribution*

Enquanto a Tabela 8 indica a informação utilizada para implementar as regras de produção, a Tabela 9 apresenta a relação entre essas regras e os atributos de transparência por elas identificados. Por exemplo, a regra *ruleIntentionality* indica que no modelo de requisitos há os atributos de transparência *rastreabilidade* e *verificabilidade*.

Tabela 9: Relação entre as regras de produção e os atributos de transparência.

Regras de produção	Atributos de transparência
ruleIntentionality	rastreabilidade e verificabilidade
ruleSoftgoal	clareza, completetude e acurácia
ruleIntentionalityDetailed	compositividade e divisibilidade
ruleAlternativeOfOperationalization	integridade, detalhamento e validade
ruleStrategicDependenceOfResource	operabilidade
ruleStrategicDependenceOfGoal	clareza, verificabilidade e rastreabilidade
ruleStrategicDependenceOfSoftgoal	compositividade
ruleTaxonomyOfActors	disponibilidade
ruleOrganizationalStructure	verificabilidade
ruleResponsibility	divisibilidade e desempenho
ruleContribution	clareza, dependência e rastreabilidade
ruleNotIntentionality	rastreabilidade e verificabilidade
ruleNotSofgoal	clareza, completeza e acurácia
ruleNotDetailedIntention	compositividade e divisibilidade
ruleNotAlternativeOfOperationalization	integridade, detalhamento e validade
ruleNotDependence	operabilidade clareza, verificabilidade e rastreabilidade compositividade
ruleNotTaxonomyOfActors	disponibilidade
ruleNotOrganizationalStructure	verificabilidade
ruleNotResponsibility	divisibilidade e desempenho
ruleNotContribution	clareza, dependência e rastreabilidade

O Código 1 mostra a implementação da regra de produção *ruleIntentionality* para verificar a presença do atributo de transparência *intencionalidade*. Nela há duas

condições necessárias para que a ação seja disparada, linhas 2 e 3. Se na base de fatos houver um elemento do tipo *actor,* outro do tipo *goal* e o *idActor* do actor for igual ao *idActor* do elemento *goal*, então ocorrerá o disparo dessa regra. Isso significa que para toda meta na base de dados ela será relacionada ao seu respectivo ator indicando assim a intencionalidade do mesmo. Quando a regra de intencionalidade é disparada dois novos fatos são adicionados à base de dados, o fato *attribute* indicando a presença do atributo *verificabilidade* e outro para indicar a presença do atributo *rastreabilidade*, conforme linhas 5 à 19.

```
1   (defrule ruleIntentionality "1 - Intentionality (actor and goal)"
2           (element (id ?idGoal)  (name ?nameGoal) (type "goal") (idActor ?idActor ) )
3           (element (id ?idActor) (name ?nameActor)                                    )
4           =>
5           (assert (attribute
6                                   (attribute "Traceability")
7                                   (rule   "1 - Intentionality (actor e goal)" )
8                                   (feature "actor and goal")
9                                   (detail (sym-cat ?nameActor " <-> " ?nameGoal))
10
11                          )
12          )
13          (assert (attribute
14                                   (attribute "Verifiability")
15                                   (rule   "1 - Intentionality (actor e goal)" )
16                                   (feature "actor and goal")
17                                   (detail (sym-cat ?nameActor " <-> " ?nameGoal))
18                  )
19          )
20  )
21
```

Código 1: A regra ruleIntentionality.

Quando um atributo é adicionado à base de fatos do sistema de verificação, é armazenada também a regra responsável por sua identificação, o conceito do *framework* utilizado na regra e também os elementos do modelo de requisitos que disparam essa regra. Permitindo assim rastrear as informações que originaram a identificação de um dado atributo na especificação de requisitos em questão. Observe esse fato nas linhas 7 à 9 e 14 à 17, do Código 1.

O iStar permite representar metas flexíveis através do elemento *softgoal,* indicando que os critérios para a satisfação do objetivo dependem do ponto de vista do ator. Segundo Oliveira et al. (2007), as metas flexíveis permitem identificar os

seguintes atributos de transparência: *clareza*, *acurácia* e *completude*. Nesse sentido, a regra *ruleSoftgoal* foi implementada conforme mostra o Código 2. A condição necessária para que a regra seja disparada é a presença de um elemento do tipo *softgoal* na base de fatos. Quando ocorre essa correspondência na base de dados, três novos fatos *attribute* são criados, os atributos de transparência *clareza*, *acurácia* e *completude*.

```
1   (defrule ruleSoftgoal "2 - Softgoal"
2           (element (id ?idSoftgoal)  (name ?nameSoftgoal) (type "softgoal") (idActor ?idActor ) )
3           =>
4           (assert (attribute
5                                   (attribute "Completeness")
6                                   (rule   "2 - Softgoal")
7                                   (feature "softgoal")
8                                   (detail ?nameSoftgoal)
9                   )
10          )
11          (assert (attribute
12                                  (attribute "Clarity")
13                                  (rule   "2 - Softgoal")
14                                  (feature "softgoal")
15                                  (detail ?nameSoftgoal)
16                  )
17          )
18          (assert (attribute
19                                  (attribute "Accuracy")
20                                  (rule   "2 - Softgoal")
21                                  (feature "softgoal")
22                                  (detail ?nameSoftgoal)
23                  )
24          )
25  )
26
```

Código 2: A regra ruleSoftgoal.

A Tabela 10 apresenta a descrição sucinta das demais regras de produção implementadas para identificar a aderência do modelo de requisitos aos atributos de transparência. O Apêndice 6 mostra a especificação dessas regras, a lógica empregada em sua implementação e os novos fatos adicionados na base de dados.

Tabela 10: Descrição das regras de produção

Regras de produção	Descrição sucinta
ruleIntentionality	Identifica a intencionalidade dos atores a partir de suas metas.
ruleSoftgoal	A partir de metas flexíveis indica os atributos *clareza, completetude* e *acurácia.*
ruleIntentionalityDetailed	Identifica a partir de metas e suas tarefas o detalhamento das intenções do ator.
ruleAlternativeOfOperationalization	Indica as alternativas de operacionalização de metas.
ruleStrategicDependenceOfResource	Sinaliza a dependência de recurso entre dois atores.
ruleStrategicDependenceOfGoal	Aponta a dependência de meta entre dois atores.
ruleStrategicDependenceOfSoftgoal	Informa a dependência de meta flexível entre dois atores.
ruleTaxonomyOfActors	Identifica a classificação dos atores.
ruleOrganizationalStructure	Informa a estrutura organizacional presente no modelo de requisitos.
ruleResponsibility	Identifica as responsabilidades dos atores.
ruleContribution	Sinaliza as contribuições positivas/negativas sobre as metas flexíveis.
ruleNotIntentionality	Indentifica a ausência de intencionalidade dos atores.
ruleNotSofgoal	Sinaliza a ausência de modelagem de metas flexíveis.
ruleNotDetailedIntention	Indentifica a ausência do detallhamento das intenções dos atores.
ruleNotAlternativeOfOperationalization	Indica a ausência no modelo das alternativas de operacionalização de metas.
ruleNotDependence	Identifica a ausência de dependência entre atores no modelo de requisitos.
ruleNotTaxonomyOfActors	Sinaliza a ausência de classificação de atores no modelo de requisitos.
ruleNotOrganizationalStructure	Identifica a ausência de modelagem da estrutura organizacional.
ruleNotResponsibility	Indica a ausência de modelagem das responsabilidades dos atores.
ruleNotContribution	Identifica a ausência da modelagem das contribuições positivas/negativas.

3.5 Sinônimos dos Atributos de Transparência

A partir da consulta à base de dados Wordnet (2012), foram identificados os sinônimos dos atributos de transparência com o objetivo de verificar se no modelo de requisitos esses termos são mencionados. A Tabela 11 contém a lista desses sinônimos e a partir dela foi criada a base de fatos contendo os atributos de transparência e seus respectivos sinônimos, conforme mostra o Apêndice 2. A partir dessa base de dados, o mecanismo de inferência do sistema CLIPS pode aplicar a regra de produção *ruleDenoteSynonym*, exibida no Código 12, para identicar no modelo de requisitos os sinônimos dos atributos de transparência. Por exemplo, quando for citada a palavra *revelation* na especificação, a regra de produção será disparada informando que há o relacionamento com o atributo de transparência *disclosure.*

A Figura 14 apresenta o *template word* usado para representar o fato *word*, responsável pelo agrupamento dos sinônimos bem com sua relação com os atributos de transparência.

```
21  (deftemplate word "The attributes"
22       (slot id           (type INTEGER) )
23       (slot word         (type STRING)  )
24       (slot idAttribute  (type INTEGER) )
25  )
```

Figura 14: Template *word* usado para representar os sinônimos.

O *slot idAttribute* é responsável pelo relacionamento entre o sinônimo e o atributo de transparência. Por exemplo, o atributo *disclosure* está relacionado aos sinônimos *revelation* e *revealing,* conforme mostra a Figura 15.

```
5       (word (id 3)    (word "disclosure") )
6       (word (id 35)  (word "revelation") (idAttribute 3))
7       (word (id 36)  (word "revealing") (idAttribute 3))
```

Figura 15: Base de fatos contendo atributos de transparência e seus sinônimos.

A relação de sinônimos de atributos de transparência listada na Tabela 11 é uma proposta inicial que pode ser melhorada a partir de sua utilização em modelos de requisitos disponíveis na literatura.

Tabela 11: Relação de atributos e seus sinônimos.
Fonte: Wordnet, 2012.

Atributos	Atributos	Relação
portável	*portable*	*portable, portability*
disponível	*available*	*availability, usable, useable, uncommitted*
publicidade	*disclosure*	*revelation, revealing*
uniforme	*uniform*	*consistent, undifferentiated, unvarying*
simples	*simplicity*	*ease, easiness, simpleness*
operável	*operable*	*operable, practicable, functional, usable, useable, operability, in working order*
intuitivo	*intuitive*	*intuitive , nonrational, visceral*
desempenho	*performance*	*execution, operation, functioning, carrying out, carrying into action*
adaptável	*adaptable*	*adaptability, adaptable*
amigável	*friendly*	*friendliness, friendly*
clareza	*clarity*	*lucidity, lucidness, pellucidity, clearness, limpidity*
completa	*complete*	*fill out, fill in, make out, completeness, full*
correto	*correct*	*right, correctness, rightness*
atualizada	*updated*	*update*
comparável	*comparable,*	*corresponding, like, comparison, compare, equivalence, comparability*
consistência	*consistency*	*consistence*
Integridade	*integrity*	*unity, wholeness*
acurácia	*accuracy*	*truth*
concisão	*conciseness*	*concision, pithiness, succinctness*
compositividade	*compound*	*compound*
divisível	*divisible*	*divisibility*
detalhamento	*detail*	*item, point, particular*
dependência	*dependence*	*dependance, dependency*
validável	*authenticate*	*authenticate*
controlável	*controllable*	*controllable, controllability, governable*
verificável	*verifiable*	*confirmable*
rastreável	*traceable*	*trackable, traced*
Explicável	*explainable*	*understandable*

4 EXEMPLO DA UTILIZAÇÃO DO SISTEMA DE VERIFICAÇÃO

Nesse capítulo é apresentado um exemplo de inferência sobre o modelo de requisitos do sistema *Patient-centred care* (assistência médica orientada ao paciente) abordado no trabalho de Yu (2001). A Figura 16 mostra o diagrama de razões estratégicas desse domínio, onde são representadas as informações sobre esse sistema de assistência à saúde. Nesse cenário os registros e históricos dos pacientes são controlados pelo próprio paciente. Um sistema de software que disponibiliza tal serviço deve conceder acesso autorizado aos profissionais de saúde para diagnosticar e tratar seus pacientes com base em suas informações. Em contraste com esse cenário, cada provedor de serviços de assistência médica gera e mantém as informações sobre seus pacientes resultando em dados fragmentados, duplicados, gerando atrasos no tratamento e custos elevados. Por exemplo, os exames de laboratório podem ser repetidos em unidades diferentes, se os dados fossem orientados ao paciente a informação de um dado exame poderia ser reutilizada por outro provedor de assistência médica.

O cenário do sistema *Patient-Centred Care*, Figura 16, mostra que o paciente deseja ter uma vida saudável, *keep well*, e os meios pelos quais ele pode alcançar esse objetivo é através das atividades *Patient-Centred Care* e *Provider-Centred Care*. Ou seja, usando um sistema de assistência médica orientada ao paciente ou através do sistema tradicional de assistência à saúde com informações centralizadas nos provedores do serviço. Quando o paciente desempenha a atividade *Patient-Centred Care* ele executa duas subtarefas: *Follow Customized Treatment Plan e Plan Life Activities.* Ele segue um plano de tratamento de acordo com suas necessidades e integrado às atividades do seu cotidiano.

O paciente deseja também manter sua privacidade, seu estilo de vida, a qualidade da assistência e rapidez no atendimento. Esses objetivos possuem critérios de avaliação subjetivos, portanto foi modelado com o construtor *softgoal*. Outros dois *softgoals* contribuem para que a privacidade seja mantida: minimizar intrusão e sigilo dos dados médicos do paciente. A tarefa restringir acesso possibilita o sigilo das informações do paciente, enquanto a tarefa *Patient-Centred Care* ajuda a alcançar os *softgoals* minimizar intrusão, economia de tempo e estilo de vida

normal. Quando o paciente utiliza um serviço de assistência à saúde orientada às suas necessidades, o tempo de atendimento é menor, conseqüentemente contribui para o *softgoal Time Saving*, as atividades em seu cotidiano são integradas ao tratamento ajudando na obtenção do *softgoal Normal LifeStile* e políticas de segurança para os seus dados pessoais são estabelecidas aumentando assim sua privacidade.

O provedor de serviços médicos, *Healthcare Provider*, deseja oferecer serviços de assistência à saúde viáveis, com eficiência operacional e tratamento eficaz de moléstias. Esses objetivos foram representados com os *softgoals: Viable Healthcare Service, Efficient Operations e Effective Treatments*. A eficiência operacional e o tratamento eficaz contribuem positivamente para que o *softgoal Viable Healthcare Service* seja alcançado. A tarefa *Patient-Centred Treatment* ajuda na obtenção do *softgoal Efficient Operations,* porém sua contribuição para o objetivo tratamento eficaz é desconhecida.

As dependências estratégicas entre os atores foram modeladas, por exemplo, o provedor de serviços de saúde depende que o paciente faça a adesão ao plano de tratamento. Nesse caso o *depender* é o *Healthcare Provider,* o *dependum* é *Adhere To Treatment Plan* e o *dependee* é o *Patient.*

O processo de identificação da presença de atributos de transparência é descrito na Figura 17. A primeira atividade realizada nesse processo é a especificação dos requisitos do sistema de software utilizando o *Framework* iStar. Esse framework descreve informações aderentes aos atributos de transparência, conforme relatado no trabalho de Oliveira et al. (2007), Leite e Cappelli (2008), ou seja, ele possui características que permitem identificar os atributos de transparência.

A partir do modelo de requisitos representado com o *Framework* iStar, Figura 16, foi formada a base de dados contendo os fatos que descrevem os elementos desse modelo, atividade 3 desse processo. Quando a ferramenta de modelagem iStar não gerar a representação do modelo em iStarML, a atividade 2 não é realizada. O Apêndice 1 mostra o diagrama da Figura 16 representado em fatos CLIPS, gerado a partir da execução da atividade 3.

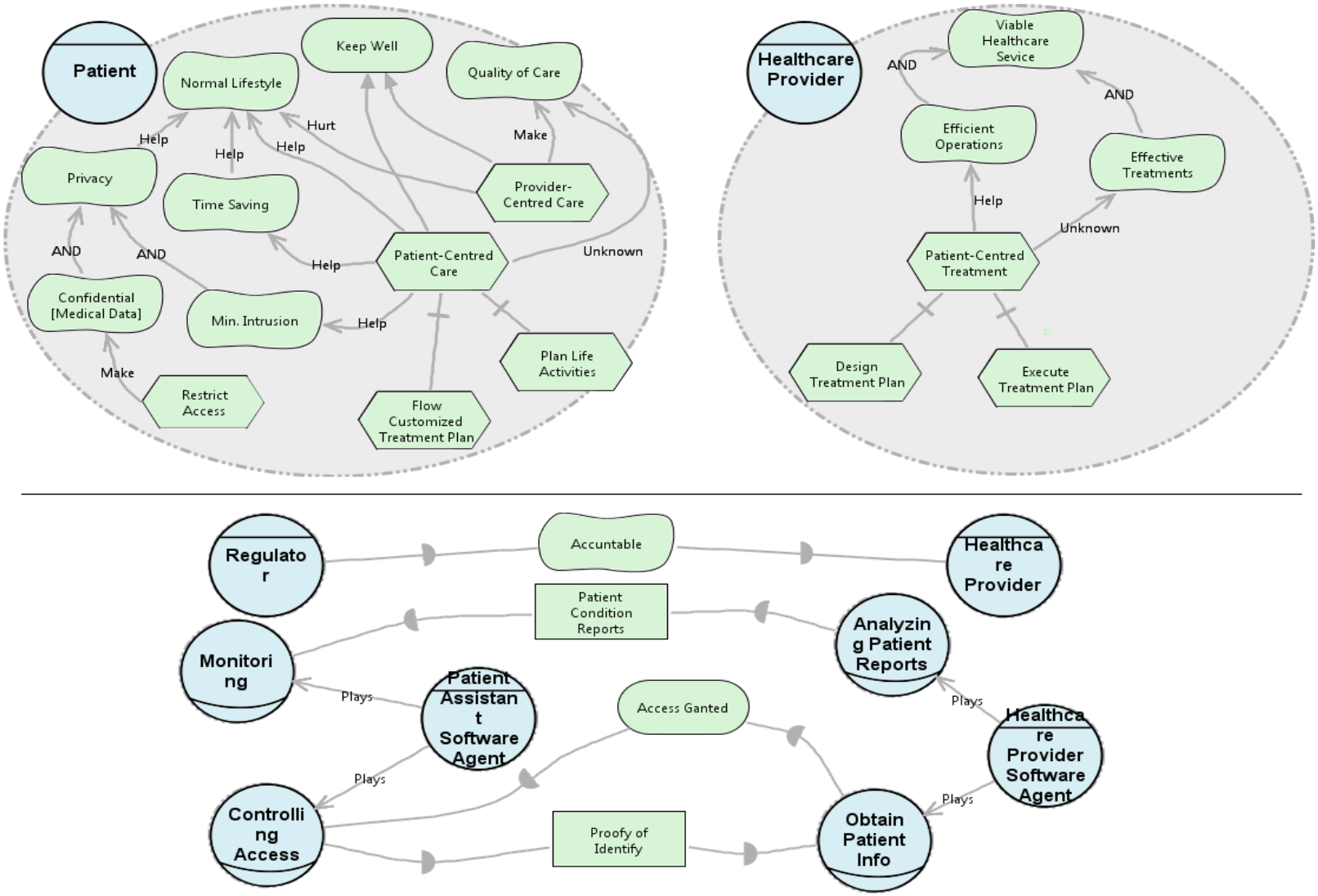

Figura 16: Modelo de razões estratégicas do domínio *Patient-Centred Care*.

Fonte: Adaptado de Yu, (2001).

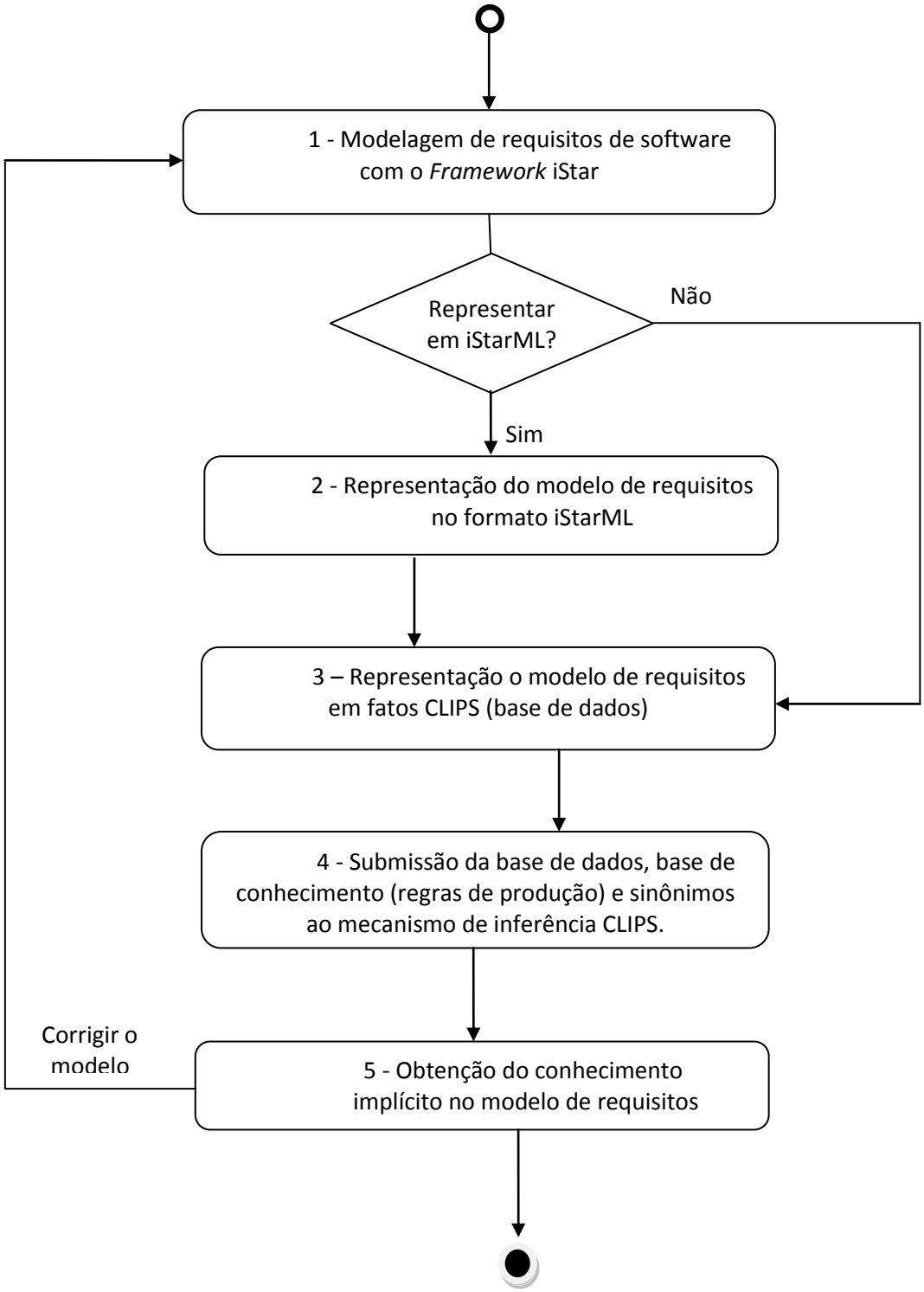

Figura 17: Atividades do processo de identificação de atributos de transparência.

Quando a atividade 4 é executada, a base de dados, os sinônimos e a base de conhecimento são submetidos ao mecanismo de inferência do sistema CLIPS. A base de dados contendo os sinônimos dos requisitos no formato de fatos está

disponível no Apêndice 2 e o Apêndice 3 contém a base de conhecimento formada pelas regras de produção.

Após o processo de inferência, os dados obtidos podem ser analisados e organizados de acordo com a necessidade do usuário, atividade 5. O analista pode utilizar as informações inferidas nessa etapa e modificar o modelo, voltando na atividade 01. Nesse exemplo as informações obtidas foram tabuladas conforme Apêndices 4 e 5. Nessa etapa é possível visualizar os atributos de transparência identificados pelas regras de produção bem como a ausência na especificação de requisitos de informações que podem indicar a aderência desse modelo aos atributos de transparência. As informações obtidas foram organizadas em três colunas. Os atributos de transparência, a regra responsável por sua identificação e os elementos do iStar que dispararam a regra são exibidos respectivamente nas colunas *Atributos de transparência*, *regras de produção* e *elementos do iStar*. Por exemplo, as duas primeiras linhas mostram que a regra *1 - Intentionality (actor e goal)* identificou os atributos de transparência verificabilidade e rastreabilidade a partir da existência do agente *HealthCare Provider Software Agent* e do *goal Conditions monitored*. O software de monitoramento das condições do paciente tem como principal função acompanhar as condições de saúde do paciente, ou seja, essa informação revela a intencionalidade desse agente e de acordo com a Tabela 4, isso indica a presença dos atributos de transparência *verificabilidade* e *rastreabilidade*.

As regras de produção também identificaram a ausência de várias informações no modelo de requisitos que poderiam sinalizar a presença dos atributos de transparência, se ali estivessem. O Apêndice 5 exibe essas informações. Por exemplo, o *goal Conditions monitored* foi mencionado no modelo sem especificar suas alternativas de operacionalização. Nesse caso a regra *Not alternative of operationalization* identificou esse fato e sinalizou que os atributos de transparência *detalhamento*, *validade* e *integridade* poderiam ser indicados caso houvesse essa informação na especificação de requisitos.

A Tabela 12 mostra o número de ocorrência de cada atributo de transparência identificado na especificação de requisitos do sistema *Patient-Centred Care*, exibido na Figura 16. Por exemplo, o atributo *Accuracy* foi indicado 15 vezes pelas regras de produção.

Tabela 12: Número de ocorrência dos atributos de transparência.

Atributos de transparência identificados	Número de ocorrência
Accuracy	15
Clarity	30
Completeness	15
Compositionality	5
Dependence	14
Detail	2
Divisibility	8
Integrity	2
Operability	2
Performance	4
Traceability	19
Validity	2
Verifiability	5

A Tabela 13 exibe a relação de atributos de transparência ausentes na especificação de requisitos do sistema *Patient-Centred Care*. A partir das características do iStar foi possível implementar regras de produção para indicar partes do modelo de requisitos que não sinalizam atributos de transparência. Por exemplo, foi indicada a ausência do atributo *Avalability* em nove pontos da especificação. Conforme mostra o Apêndice 5, a regra *ruleNotTaxonomyOfActors* identificou a falta de classificação para os atores: *HealthCare Provider Software Agent, Analizing Patient Reports, Obtain Patient Info, Monitoring, Controlling Access, Patient Assistant Software Agent, Healthcare Provider, Regulator* e *Patient.* Conforme a Tabela 9, essa regra indica a ausência do atributo de transparência *disponibilidade*.

Tabela 13: Atributos ausentes no sistema *Patient-Centred Care*.

Atributos de transparência ausentes	Número de ocorrência
Availability	9
Clarity	11
Compositionality	12
Dependence	6
Detail	3
Divisibility	14
Integrity	3
Operability	5
Performance	7
Traceability	16
Validity	3
Verifiability	19

Conforme apresentado na Tabela 4, existem atributos de transparência que não estão relacionados aos conceitos do iStar. Portanto não foi possível implementar regras de produção para indicar a presença ou ausência desses atributos em especificações de requisitos representadas com o iStar.

Nesse capítulo foi executado o processo de verificação de atributos de transparência na especificação de requisitos do sistema *Patient-Centred Care*, exibido Figura 16. Os requisitos foram transformados em fatos para formar a base de dados. A partir da base de conhecimento desenvolvida foi possível identificar a presença dos requisitos listados na Tabela 12. Além disso, foram indicados pontos nessa especificação onde há a ausência dos atributos de transparência, conforme exibido na Tabela 13. Além disso, no modelo de requisitos da Figura 16, não foi citado nenhum sinônimo dos atributos de transparência listados na Tabela 4, portanto a regra *ruleDenoteSynonym* não sinalizou essa ocorrência.

5 CONCLUSÕES E PERSPECTIVAS FUTURAS

A transparência de *software* é um requisito que engenheiros de *software* precisarão demonstrar à medida que as pessoas exigirem transparência em suas relações com a sociedade, pois essas relações são automatizadas por sistemas de *software*. A partir dessa perspectiva, nesse trabalho foi desenvolvida uma base de conhecimento para verificar a aderência de especificações de requisitos de *software* aos atributos de qualidade que contribuem para alcançar a transparência de *software*. A partir dos conceitos disponíveis no *Framework* iStar foi possível desenvolver regras de produção para realizar essa verificação.

A estratégia desenvolvida para identificar atributos de transparência em especificações de requisitos representadas com o iStar utiliza o sistema de regras produção CLIPS para implementar a base de conhecimento, representar o modelo de requisitos e para realizar a inferência do conhecimento implícito na base de fatos. Essa abordagem baseada em sistema de inferência permitiu a obtenção de um sistema flexível e expansível, que proporciona seu uso para verificar especificações de requisitos e para sugerir a inclusão de novos atributos em especificações analisadas pelo sistema. Essa estratégia permite também o uso do sistema como uma plataforma de ensino e disseminação das ideias de transparência e sua inclusão em sistemas de informação organizacionais.

A separação entre a base de dados contendo a especificação de requisitos e as regras de produção proporcionou a obtenção de um sistema de verificação composto por módulos independentes. Novas regras de produção podem ser adicionadas à medida que o sistema evolui. A base de conhecimento desenvolvida nesse trabalho pode ser utilizada por um agente inteligente baseado em conhecimento, além disso, pode ser integrada às ferramentas de modelagem iStar.

Além de utilizar os conceitos do *Framework* iStar para implementar a base de conhecimento, foram utilizados sinônimos dos atributos de transparência para indicar a presença de palavras na especificação de requisitos que indicam os atributos de transparência. Essa relação de sinônimos é uma proposta inicial que pode ser melhorada a partir de sua utilização em modelos de requisitos disponíveis na literatura.

A partir da especificação de requisitos do sistema *Patient-Centred Care* representada com o *Framework* iStar, Figura 16, foi possível observar o funcionamento do sistema de verificação. A base de conhecimento submetida ao mecanismo de inferência do CLIPS permitiu extrair o conhecimento implícito nessa especificação de requisitos, conforme apresentado nos Apêndices 4 e 5.

5.1 Trabalhos futuros

Como trabalho futuro sugere-se a elaboração de uma lista de verbos que são indícios sinalizadores da necessidade de atributos de transparência em especificações de requisitos de *software*. A partir dessa listagem construir a base de dados semelhante à base de fatos contendo os sinônimos apresentados no Apêndice 2. Outra sugestão é integrar a base de conhecimento desenvolvida nesse trabalho às ferramentas de modelagem de requisitos iStar com o objetivo de automatizar o processo de verificação. Por fim, sugere-se adotar um mecanismo de classificação que indique o grau de transparência presente na especificação de requisitos analisada.

APÊNDICE 1: Base de Dados Contendo os Fatos do Sistema *Patient-Centred Care*

```
(deffacts requirements "Patient-Centred Care"

(element (id "01") (name "Patient")                              (type "actor"))
(element (id "02") (name "Regulator")                            (type "actor"))
(element (id "03") (name "Healthcare Provider")                  (type "actor"))
(element (id "04") (name "Patient Assistant Software Agent")     (type "actor"))
(element (id "05") (name "Controlling Access                     (type "actor"))
(element (id "06") (name "Monitoring")                           (type "actor"))
(element (id "07") (name "Obtain Patient Info")                  (type "actor"))
(element (id "08") (name "Analizing Patient Reports")            (type "actor"))
(element (id "09") (name "HealthCare Provider Software Agent") (type "actor"))

(element (id "10") (name "Privacy")                          (type "softgoal") (idActor "01"))
(element (id "11") (name "Normal Lifestyle")                 (type "softgoal") (idActor "01"))
(element (id "12") (name "Quality of Care")                  (type "softgoal") (idActor "01"))
(element (id "13") (name "Time Saving")                      (type "softgoal") (idActor "01"))
(element (id "14") (name "Min. Intrusion")                   (type "softgoal") (idActor "01"))
(element (id "15") (name "Confidencial Medical Data")        (type "softgoal") (idActor "01"))
(element (id "16") (name "Accommodating Daily Adjustments")(type "softgoal") (idActor "01"))
(element (id "17") (name "Legitimate User Only")             (type "softgoal") (idActor "05"))
(element (id "18") (name "Adhere To Treatment Plan")         (type "softgoal") (idActor "01"))
(element (id "19") (name "Flexible Treatment Plan")          (type "softgoal") (idActor "01"))
(element (id "20") (name "Trustworthy [Healthcare System]")(type "softgoal") (idActor "01"))
(element (id "21") (name "Accountable")                      (type "softgoal") (idActor "03"))
(element (id "22") (name "Viable Healthcare Service")        (type "softgoal") (idActor "03"))
(element (id "23") (name "Efficient Operations")             (type "softgoal") (idActor "03"))
(element (id "24") (name "Effective Treatment")              (type "softgoal") (idActor "03"))

(element (id "25") (name "Keep Well")                 (type "goal") (idActor "01"))
(element (id "26") (name "Vital Signs Monitored")     (type "goal") (idActor "06"))
(element (id "27") (name "Access Ganted [Request]")   (type "goal") (idActor "05"))
(element (id "28") (name "Conditions monitored")      (type "goal") (idActor "09"))

(element (id "29") (name "Patient Centred Care")             (type "task") (idActor "01"))
(element (id "30") (name "Provider Centred Care")            (type "task") (idActor "01"))
(element (id "31") (name "Restrict Access")                  (type "task") (idActor "01"))
(element (id "32") (name "Flow Customized Treatment Pla")   (type "task") (idActor "01"))
```

```
(element (id "33") (name "Plan Life Activities")           (type "task") (idActor "01"))
(element (id "34") (name "Customize Treatment Plan")       (type "task") (idActor "04"))
(element (id "35") (name "Patient Centred Treatment")      (type "task") (idActor "03"))
(element (id "36") (name "Design Treatment Plan")          (type "task") (idActor "03"))
(element (id "37") (name "Execute Treatment Plan")         (type "task") (idActor "03"))

(element (id "38") (name "Customized Treatment Plan")(type "resource") (idActor "04"))
(element (id "39") (name "Lifestyle Preferences")     (type "resource") (idActor "04"))
(element (id "40") (name "Personal Medical Data")     (type "resource") (idActor "04"))
(element (id "41") (name "Log Access")                (type "resource") (idActor "05"))
(element (id "42") (name "Treatment Plan")            (type "resource") (idActor "04"))
(element (id "43") (name "Patient Condition Reports")(type "resource") (idActor "04"))
(element (id "44") (name "Patient Data")              (type "resource") (idActor "09"))
(element (id "45") (name "Proof Identity")            (type "resource") (idActor "05"))

(elementLink (type "means-end")     (source "29") (target "25") )
(elementLink (type "means-end")     (source "30") (target "25") )
(elementLink (type "decomposition") (source "32") (target "29") )
(elementLink (type "decomposition") (source "33") (target "29") )
(elementLink (type "decomposition") (source "36") (target "35") )
(elementLink (type "decomposition") (source "37") (target "35") )

(elementLink (type "contribution")  (label "Help")     (source "13") (target "11") )
(elementLink (type "contribution")  (label "Help")     (source "29") (target "11") )
(elementLink (type "contribution")  (label "Hurt")     (source "30") (target "11") )
(elementLink (type "contribution")  (label "Unknown")  (source "29") (target "12") )
(elementLink (type "contribution")  (label "Make")     (source "30") (target "12") )
(elementLink (type "contribution")  (label "Help")     (source "29") (target "13") )
(elementLink (type "contribution")  (label "Help")     (source "29") (target "14") )
(elementLink (type "contribution")  (label "And")      (source "14") (target "10") )
(elementLink (type "contribution")  (label "And")      (source "15") (target "10") )
(elementLink (type "contribution")  (label "Make")     (source "31") (target "15") )
(elementLink (type "contribution")  (label "And")      (source "23") (target "22") )
(elementLink (type "contribution")  (label "And")      (source "24") (target "22") )
(elementLink (type "contribution")  (label "Help")     (source "35") (target "23") )
(elementLink (type "contribution")  (label "Unknown")  (source "35") (target "24") )

(elementLink (type "dependence") (source "02") (target "21") )
(elementLink (type "dependence") (source "21") (target "03") )
(elementLink (type "dependence") (source "08") (target "43") )
```

```
(elementLink (type "dependence") (source "43") (target "06") )
(elementLink (type "dependence") (source "07") (target "27") )
(elementLink (type "dependence") (source "27") (target "05") )

(elementLink (type "dependence") (source "05") (target "45") )
(elementLink (type "dependence") (source "45") (target "07") )

(elementLink (type "plays") (source "04") (target "05") )
(elementLink (type "plays") (source "04") (target "06") )
(elementLink (type "plays") (source "09") (target "07") )
(elementLink (type "plays") (source "09") (target "08") )
```

APÊNDICE 2: Base de Dados de Sinônimos dos Atributos de Transparência

```
(deffacts words
      (word (id 1)   (word "portable") )
      (word (id 2)   (word "available") )
      (word (id 3)   (word "disclosure") )
      (word (id 4)   (word "uniform") )
      (word (id 5)   (word "simplicity") )
      (word (id 6)   (word "operable") )
      (word (id 7)   (word "intuitive") )
      (word (id 8)   (word "performance") )
      (word (id 9)   (word "adaptable") )
      (word (id 10)  (word "friendly") )
      (word (id 11)  (word "clarity") )
      (word (id 12)  (word "complete") )
      (word (id 13)  (word "correct") )
      (word (id 14)  (word "updated") )
      (word (id 15)  (word "comparable") )
      (word (id 16)  (word "consistency") )
      (word (id 17)  (word "integrity") )
      (word (id 18)  (word "accuracy") )
      (word (id 19)  (word "conciseness") )
      (word (id 20)  (word "compound") )
      (word (id 21)  (word "divisible") )
      (word (id 22)  (word "detail") )
      (word (id 23)  (word "dependence") )
      (word (id 24)  (word "authenticate") )
      (word (id 25)  (word "controllable") )
      (word (id 26)  (word "verifiable") )
      (word (id 27)  (word "traceable") )
      (word (id 28)  (word "explainable") )

      (word (id 30) (word "portability")              (idAttribute 1))
      (word (id 31) (word "availability")             (idAttribute 2))
      (word (id 32) (word "usable")                   (idAttribute 2))
      (word (id 33) (word "useable")                  (idAttribute 2))
      (word (id 34) (word "uncommitted")              (idAttribute 2))
      (word (id 35) (word "revelation")               (idAttribute 3))
      (word (id 36) (word "revealing")                (idAttribute 3))
      (word (id 37) (word "consistent")               (idAttribute 4))
      (word (id 38) (word "undifferentiated")         (idAttribute 4))
      (word (id 39) (word "unvarying")                (idAttribute 4))
      (word (id 40) (word "ease")                     (idAttribute 5))
      (word (id 41) (word "easiness")                 (idAttribute 5))
      (word (id 42) (word "simpleness")               (idAttribute 5))
      (word (id 43) (word "operable")                 (idAttribute 6))
      (word (id 44) (word "practicable")              (idAttribute 6))
      (word (id 45) (word "functional")               (idAttribute 6))
      (word (id 46) (word "usable")                   (idAttribute 6))
      (word (id 47) (word "useable")                  (idAttribute 6))
      (word (id 48) (word "operability")              (idAttribute 6))
      (word (id 49) (word "in working order")         (idAttribute 6))
      (word (id 50) (word "intuitive")                (idAttribute 7))
      (word (id 51) (word "nonrational")              (idAttribute 7))
      (word (id 52) (word "visceral")                 (idAttribute 7))
      (word (id 53) (word "execution")                (idAttribute 8))
      (word (id 54) (word "operation")                (idAttribute 8))
      (word (id 55) (word "functioning")              (idAttribute 8))
      (word (id 56) (word "carrying out")             (idAttribute 8))
```

```
        (word (id 57) (word "carrying into action")     (idAttribute 8))
        (word (id 58) (word "adaptability")             (idAttribute 9))
        (word (id 59) (word "adaptable")                (idAttribute 9))
        (word (id 60) (word "friendliness")             (idAttribute 10))
        (word (id 61) (word "friendly")                 (idAttribute 10))
        (word (id 62) (word "lucidity")                 (idAttribute 11))
        (word (id 63) (word "lucidness")                (idAttribute 11))
        (word (id 64) (word "pellucidity")              (idAttribute 11))
        (word (id 65) (word "clearness")                (idAttribute 11))
        (word (id 66) (word "limpidity")                (idAttribute 11))
        (word (id 67) (word "fill out")                 (idAttribute 12))
        (word (id 68) (word "fill in")                  (idAttribute 12))
        (word (id 69) (word "make out")                 (idAttribute 12))
        (word (id 70) (word "completeness")             (idAttribute 12))
        (word (id 71) (word "full")                     (idAttribute 12))
        (word (id 72) (word "right")                    (idAttribute 13))
        (word (id 73) (word "correctness")              (idAttribute 13))
        (word (id 74) (word "rightness")                (idAttribute 13))
        (word (id 75) (word "update")                   (idAttribute 14))
        (word (id 76) (word "corresponding")            (idAttribute 15))
        (word (id 77) (word "like")                     (idAttribute 15))
        (word (id 78) (word "comparison")               (idAttribute 15))
        (word (id 79) (word "compare")                  (idAttribute 15))
        (word (id 80) (word "equivalence")              (idAttribute 15))
        (word (id 81) (word "comparability")            (idAttribute 15))
        (word (id 82) (word "consistence")              (idAttribute 16))
        (word (id 83) (word "unity")                    (idAttribute 17))
        (word (id 84) (word " wholeness")               (idAttribute 17))
        (word (id 85) (word "truth")                    (idAttribute 18))
        (word (id 86) (word "concision")                (idAttribute 19))
        (word (id 87) (word "pithiness")                (idAttribute 19))
        (word (id 88) (word "succinctness")             (idAttribute 19))
        (word (id 89) (word "compound")                 (idAttribute 20))
        (word (id 90) (word "divisibility")             (idAttribute 21))
        (word (id 91) (word "item")                     (idAttribute 22))
        (word (id 92) (word "point")                    (idAttribute 22))
        (word (id 93) (word "particular")               (idAttribute 22))
        (word (id 94) (word "dependance")               (idAttribute 23))
        (word (id 95) (word "dependency")               (idAttribute 23))
        (word (id 96) (word "authenticate")             (idAttribute 24))
        (word (id 97) (word "controllable")             (idAttribute 25))
        (word (id 98) (word "controllability")          (idAttribute 25))
        (word (id 99) (word "governable")               (idAttribute 25))
        (word (id 100) (word "confirmable")             (idAttribute 26))
        (word (id 101) (word "trackable")               (idAttribute 27))
        (word (id 102) (word "traced")                  (idAttribute 27))
        (word (id 103) (word "understandable")          (idAttribute 28))
)
```

APÊNDICE 3: Base de Conhecimento

```
;;;;;;;;;;; Templates  ;;;;;;;;;;;;;;;;;;;;

(deftemplate element "The element of iStar"
       (slot id      (type STRING) )
       (slot name    (type STRING) )
       (slot type    (type STRING) )
       (slot idActor (type STRING) )
)

(deftemplate elementLink "The link of elements"
       (slot id      (type STRING) )
       (slot label   (type STRING) )
       (slot type    (type STRING) )
       (slot source  (type STRING) )
       (slot target  (type STRING) )
)

(deftemplate word "The attributes"
       (slot id              (type INTEGER) )
       (slot word            (type STRING)  )
       (slot idAttribute     (type INTEGER) )
)

(deftemplate attribute "Attributes checked"
       (slot attribute (type STRING) )
       (slot rule      (type STRING) )
       (slot feature   (type STRING) )
       (slot detail    (type STRING) )
)
```

```
;;;;;;;;;;; Rules  ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(defrule ruleIntentionality "1 - Intentionality (actor e goal)"
      (element (id ?idGoal)  (name ?nameGoal) (type "goal") (idActor ?idActor ) )
      (element (id ?idActor) (name ?nameActor))
      =>
      (assert (attribute
                        (attribute "Traceability")
                        (rule   "1 - Intentionality (actor e goal)" )
                        (feature "actor and goal")
                        (detail (sym-cat ?nameActor " <-> " ?nameGoal))      )           )
      (assert (attribute
                              (attribute "Verifiability")
                              (rule   "1 - Intentionality (actor e goal)" )
                              (feature "actor and goal")
                              (detail (sym-cat ?nameActor " <-> " ?nameGoal))      )           )                 )

(defrule ruleSoftgoal "2 - Softgoal"
      (element (id ?idSoftgoal)  (name ?nameSoftgoal) (type "softgoal") (idActor ?idActor ) )
      =>
      (assert (attribute
                        (attribute "Completeness")
                        (rule   "2 - Softgoal")
                        (feature "softgoal")
                        (detail ?nameSoftgoal)  )     )
      (assert (attribute
                        (attribute "Clarity")
                        (rule   "2 - Softgoal")
                        (feature "softgoal")
                        (detail ?nameSoftgoal)    )         )
      (assert (attribute
                        (attribute "Accuracy")
                        (rule   "2 - Softgoal")
                        (feature "softgoal")
                        (detail ?nameSoftgoal)    )         )           )
```

```
(defrule ruleIntentionalityDetailed "3 - Intentionality detailed (task decomposition)"
      (element        (id ?idTask)     (name ?nameTask)   (type "task") )
      (elementLink    (label ?label)   (type "decomposition")  (source ?source) (target ?idTask) )
      (element        (id ?source)     (name ?nameElement) )
      =>
      (assert (attribute
                       (attribute "Compositionality")
                       (rule   "3 - Intentionality detailed (task decomposition)")
                       (feature "decomposition link")
                       (detail (sym-cat ?nameTask " <-> " ?nameElement))          )     )
      (assert (attribute
                       (attribute "Divisibility")
                       (rule   "3 - Intentionality detailed (task decomposition)")
                       (feature "decomposition link")
                       (detail (sym-cat ?nameTask " <-> " ?nameElement))    )     )           )

(defrule ruleAlternativeOfOperationalization "4 - Alternative of operationalization (means-end link)"
      (element     (id ?idGoal)    (name ?nameGoal)    (type "goal")      (idActor ?idActor ) )
      (element     (id ?idElement) (name ?nameElement) (type ?typeElement)(idActor ?idActor ) )
      (elementLink (type "means-end")  (source ?idElement)   (target ?idGoal) )
      =>
      (assert (attribute
                             (attribute "Integrity")
                             (rule   "4 - Alternative operationalization (means-end link)")
                             (feature "means-end link")
                             (detail (sym-cat ?nameGoal " <-> " ?nameElement))    )     )
      (assert (attribute
                             (attribute "Detailing")
                             (rule   "4 - Alternative operationalization (means-end link)")
                             (feature "means-end link")
                             (detail (sym-cat ?nameGoal " <-> " ?nameElement))    )     )
      (assert (attribute
                             (attribute "Validity")
                             (rule   "4 - Alternative operationalization (means-end link)")
                             (feature "means-end link")
                             (detail (sym-cat ?nameGoal " <-> " ?nameElement))    )     )           )
```

```
(defrule ruleStrategicDependenceOfGoal "5.2 - Strategic dependence of goal"
      (elementLink    (type "dependence")  (source ?idDepender) (target ?idElement))
      (element        (id ?idDepender)     (name ?nameDepender) (type "actor")  )
      (element        (id ?idElement)      (name ?nameElement)  (type "goal")    (idActor ?idActor ) )
      (elementLink    (type "dependence")  (source ?idElement)  (target ?idDependee) )
      (element        (id ?idDependee)     (name ?nameDependee) (type "actor") )
      =>
      (assert (attribute
                           (attribute "Clarity")
                           (rule   "5.2 - Strategic dependence of goal")
                           (feature "dependence link, goal")
                           (detail (sym-cat ?nameDepender " <-> " ?nameElement " <-> " ?nameDependee))
                )
      )
      (assert (attribute
                           (attribute "Verifiability")
                           (rule   "5.2 - Strategic dependence of goal")
                           (feature "dependence link, goal")
                           (detail (sym-cat ?nameDepender " <-> " ?nameElement " <-> " ?nameDependee))
                )
      )
      (assert (attribute
                           (attribute "Traceability")
                           (rule   "5.2 - Strategic dependence of goal")
                           (feature "dependence link, goal")
                           (detail (sym-cat ?nameDepender " <-> " ?nameElement " <-> " ?nameDependee))
                )
      )

)
```

```
(defrule ruleStrategicDependenceOfResource "5.1 - Strategic dependence of resource"
      (elementLink    (type "dependence")  (source ?idDepender) (target ?idElement) )
      (element        (id ?idDepender)     (name ?nameDepender) (type "actor")  )
      (element        (id ?idElement)      (name ?nameElement)  (type "resource")  (idActor ?idActor ) )
      (elementLink    (type "dependence")  (source ?idElement)  (target ?idDependee)  )
      (element        (id ?idDependee)     (name ?nameDependee) (type "actor") )
      =>
      (assert (attribute
                           (attribute "Operability")
                           (rule   "5.1 - Strategic dependence of resource")
                           (feature "dependence link, resource")
                           (detail (sym-cat ?nameDepender " <-> " ?nameElement " <-> " ?nameDependee))
                )
      )

)



(defrule ruleStrategicDependenceOfSoftgoal "5.3 - Strategic dependence of softgoal"
      (elementLink    (type "dependence")  (source ?idDepender) (target ?idElement))
      (element        (id ?idDepender)     (name ?nameDepender) (type "actor"))
      (element        (id ?idElement)      (name ?nameElement)  (type "softgoal")    (idActor ?idActor ) )
      (elementLink    (type "dependence")  (source ?idElement)  (target ?idDependee) )
      (element        (id ?idDependee)     (name ?nameDependee) (type "actor"))
      =>
      (assert (attribute
                           (attribute "Compositionality")
                           (rule   "5.3 - Strategic dependence of softgoal")
                           (feature "dependence link, softgoal")
                           (detail (sym-cat ?nameDepender " <-> " ?nameElement " <-> " ?nameDependee))
                )
      )

)
```

```
(defrule ruleTaxonomyOfActors "6 - Taxonomy of actors (actors link INS e ISA)"
      (elementLink (label "is_a" | "INS") (source ?sourceActorLink)  (target ?targetActorLink)  )
      (element     (id ?sourceActorLink)  (name ?nameActor) )
      (element     (id ?targetActorLink)  (name ?nameAgent) )
      =>
      (assert (attribute
                              (attribute "Availability")
                              (rule   "6 - Taxonomy of actors (actors link INS e ISA)" )
                              (feature "actors link, actor, agent")
                              (detail (sym-cat ?nameActor " <-> "?nameAgent ))
                  )
      )
)



(defrule ruleOrganizationalStructure "7 - The organizational structure (position)"
  (elementLink (label "occupies")(type ?typeActorLink)(source ?sourceActorLink) (target ?targetActorLink))
  (element     (id ?sourceActorLink)    (name ?nameActor) )
  (element     (id ?targetActorLink)    (name ?nameElementPosition) )
   =>
      (assert (attribute
                              (attribute "Verifiability")
                              (rule   "7 - The organizational structure (position)")
                              (feature "actors link (occupies)")
                              (detail (sym-cat ?nameActor " <-> " ?nameElementPosition))
                  )
      )
)
```

```
(defrule ruleResponsibility "8 - Responsibility(role)"
      (elementLink (label "plays")  (type ?typeActorLink) (source ?sourceActorLink) (target ?targetActorLink))
      (element     (id ?sourceActorLink)    (name ?nameActor))
      (element     (id ?targetActorLink)    (name ?nameElementRole) )
      =>
      (assert (attribute
                           (attribute "Divisibility")
                           (rule  "8 - Responsibility(role)")
                           (feature "actors link (plays)")
                           (detail (sym-cat  ?nameActor " <-> " ?nameElementRole))
                )
      )
      (assert (attribute
                           (attribute "Performance")
                           (rule  "8 - Responsibility(role)")
                           (feature "actors link (plays)")
                           (detail (sym-cat  ?nameActor " <-> " ?nameElementRole))
                )
      )
)



(defrule ruleDenoteSynonym
      (element (id ?id) (name ?name) (type ?type) )                    ; o elemento
      (word (id ?idAttribute) (word ?nameAttribute) )                  ; o atributo
      (word (id ?idSim) (word ?name) (idAttribute ?idAttribute))       ; o sinônimo
      =>
      (printout t ?name " denote " ?nameAttribute crlf)
)
```

```
(defrule ruleContribution "9 - Contribution +-"
      (elementLink (label ?label) (type "contribution")  (source ?source) (target ?target)  )
      (element     (id ?source)   (name ?nameElement)  )
      (element     (id ?target)   (name ?nameSoftgoal) )
      =>
      (assert (attribute
                              (attribute "Clarity")
                              (rule   "9 - Contribution +-")
                              (feature "contribution +-")
                              (detail (sym-cat ?nameElement " <-> " ?nameSoftgoal))
                  )
      )
      (assert (attribute
                              (attribute "Dependence")
                              (rule   "9 - Contribution +-")
                              (feature "contribution +-")
                              (detail (sym-cat ?nameElement " <-> " ?nameSoftgoal))
                  )
      )
      (assert (attribute
                              (attribute "Traceability")
                              (rule   "9 - Contribution +-")
                              (feature "contribution +-")
                              (detail (sym-cat ?nameElement " <-> " ?nameSoftgoal))
                  )
      )
)
```

```
(defrule ruleNotIntentionality "Indicates the lack intentionality"
      (element  (id ?idActor)  (name ?nameActor) (type "actor") )
      (not (element (id ?idGoal)  (name ?nameGoal) (type "goal") (idActor ?idActor )  )  )
      =>
      (assert (attribute
                        (attribute "Verifiability")
                        (rule    "Not intentionality")
                        (feature "actor and goal")
                        (detail  ?nameActor)    )     )
      (assert (attribute
                        (attribute "Traceability")
                        (rule    "Not intentionality")
                        (feature "actor and goal")
                        (detail  ?nameActor)    )          )                )


(defrule ruleNotSofgoal "Indicates the lack softgoal"
      (not (element (id ?idGoal)  (name ?nameGoal) (type "softgoal") (idActor ?idActor )  )  )
      =>
      (assert (attribute
                        (attribute "Completeness")
                        (rule    "Not softgoal")
                        (feature "softgoal")         )          )
      (assert (attribute
                       (attribute "Clarity")
                        (rule    "Not softgoal")
                        (feature "softgoal")         )          )
      (assert (attribute
                        (attribute "Accuracy")
                        (rule    "Not softgoal")
                        (feature "softgoal")    )     )     )
```

```
(defrule ruleNotDetailedIntention "Indicates the lack of detailed intentionality"
    (element (id ?idTask)  (name ?nameTask) (type "task") )
    (not (elementLink (label ?label) (type "decomposition")  (source ?source) (target ?idTask) ) )
      =>
      (assert (attribute
                        (attribute "Divisibility")
                        (rule    "Not detailed intention")
                        (feature "decomposition link")
                        (detail  ?nameTask)
                )
      )
      (assert (attribute
                        (attribute "Compositionality")
                        (rule    "Not detailed intention")
                        (feature "decomposition link")
                        (detail  ?nameTask)
                )
      )

)
```

```
(defrule ruleNotAlternativeOfOperationalization "Indicates the lack of alternative of operationalization"
    (element     (id ?idGoal)    (name ?nameGoal)    (type "goal") (idActor ?idActor ) )
    (not (elementLink (type "means-end")  (source ?idElement)   (target ?idGoal) ) )
      =>
      (assert (attribute
                        (attribute "Integrity")
                        (rule   "Not alternative of operationalization")
                        (feature "means-end link")
                        (detail  ?nameGoal)
                )
      )
      (assert (attribute
                        (attribute "Validity")
                        (rule   "Not alternative of operationalization")
                        (feature "means-end link")
                        (detail  ?nameGoal)
                )
      )
      (assert (attribute
                        (attribute "Detail")
                        (rule   "Not alternative of operationalization")
                        (feature "means-end link")
                        (detail  ?nameGoal)
                )
      )

)
```

```
(defrule ruleNotDependence "Indicates the lack of dependence"
      (element  (id ?idDepender)  (name ?nameDepender) (type "actor") )
      (not (elementLink   (type "dependence")  (source ?idDepender) (target ?idElement) ) )
     =>
      (assert (attribute
                        (attribute "Operability")
                        (rule   "Not dependence resource")
                        (feature "dependence link")
                        (detail  ?nameDepender)             )     )
      (assert (attribute
                        (attribute "Clarity")
                        (rule   "Not dependence of goal")
                        (feature "dependence link")
                        (detail  ?nameDepender)             )     )
      (assert (attribute
                        (attribute "Verifiability")
                        (rule   "Not dependence of goal ")
                        (feature "dependence link")
                        (detail  ?nameDepender)             )     )
      (assert (attribute
                        (attribute "Traceability")
                        (rule   "Not dependence of goal")
                        (feature "dependence link")
                        (detail  ?nameDepender)             )     )
      (assert (attribute
                        (attribute "Compositionality")
                        (rule   "Not dependence of softgoal")
                        (feature "dependence link")
                        (detail  ?nameDepender)             )     )
)
```

```
(defrule ruleNotTaxonomyOfActors "Indicates the lack of taxonomy of actors"
      (element (id ?sourceActorLink)  (name ?nameActor) (type "actor") )
      (not (elementLink (type "is_a" | "INS")  (source ?sourceActorLink)  (target ?targetActorLink) ) )
      =>
      (assert (attribute
                              (attribute "Availability")
                              (rule   "Not Taxonomy of actor")
                              (feature "links is-a and INS")
                              (detail  ?nameActor)
                  )
      )
)
```

```
(defrule ruleNotOrganizationalStructure "Indicates the lack of organizational structure"
 (element (id ?sourceActorLink)  (name ?nameActor) (type "actor") )
 (not (elementLink (label "occupies")(type ?typeActorLink)(source ?sourceActorLink)(target ?targetActorLink)) )
  =>
      (assert (attribute
                              (attribute "Verifiability")
                              (rule   "Not organizational structure (position)")
                              (feature "link (occupies)")
                              (detail  ?nameActor)
                  )
      )
)
```

```
(defrule ruleNotResponsibility "Indicates the lack of responsibility of actors"
 (element (id ?sourceActorLink)  (name ?nameActor) (type "actor"))
 (not (elementLink (label "plays")(type ?typeActorLink)(source ?sourceActorLink)(target ?targetActorLink)))
  =>
      (assert (attribute
                              (attribute "Divisibility")
                              (rule   "Not Responsibility (role)")
                              (feature "link (plays)")
                              (detail  ?nameActor)
                  )
      )
      (assert (attribute
                              (attribute "Performance")
                              (rule   "Not Responsibility (role)")
                              (feature "link (plays)")
                              (detail  ?nameActor)
               )
      )

)
```

```
(defrule ruleNotContribution "Indicates the lack of contributions +-"
    (element (id ?target)  (name ?nameSoftgoal) (type "softgoal")  )
    (not (elementLink (label ?label) (type "contribution")(source ?source) (target ?target) ) )
      =>
      (assert (attribute
                            (attribute "Clarity")
                            (rule   "Not Contribution +-")
                            (feature "contribution +-")
                            (detail  ?nameSoftgoal)
                )
      )
      (assert (attribute
                            (attribute "Dependence")
                            (rule   "Not Contribution +-")
                            (feature "contribution +-")
                            (detail  ?nameSoftgoal)
                )
      )
      (assert (attribute
                            (attribute "Traceability")
                            (rule   "Not Contribution +-")
                            (feature "contribution +-")
                            (detail  ?nameSoftgoal)
                )
      )
)


(defrule resultSet
      (attribute (attribute ?attribute) (rule ?rule) (feature ?feature) (detail ?detail))
      =>
      (printout t ?attribute "|" ?rule "|" ?detail crlf)
)
```

APÊNDICE 4: Atributos de Transparência Identificados no Sistema *Patient-Centred Care*

Atributo de transparência	**Regra de produção**	**Elementos do iStar**
Verifiability	1 - Intentionality (actor e goal)	HealthCare Provider Software Agent <-> Conditions monitored
Traceability	1 - Intentionality (actor e goal)	HealthCare Provider Software Agent <-> Conditions monitored
Verifiability	1 - Intentionality (actor e goal)	Controlling Access <-> Access Ganted [Request]
Traceability	1 - Intentionality (actor e goal)	Controlling Access <-> Access Ganted [Request]
Verifiability	1 - Intentionality (actor e goal)	Monitoring <-> Vital Signs Monitored
Traceability	1 - Intentionality (actor e goal)	Monitoring <-> Vital Signs Monitored
Verifiability	1 - Intentionality (actor e goal)	Patient <-> Keep Well
Traceability	1 - Intentionality (actor e goal)	Patient <-> Keep Well
Accuracy	2 - Softgoal	Effective Treatment
Clarity	2 - Softgoal	Effective Treatment
Completeness	2 - Softgoal	Effective Treatment
Accuracy	2 - Softgoal	Efficient Operations
Clarity	2 - Softgoal	Efficient Operations
Completeness	2 - Softgoal	Efficient Operations
Accuracy	2 - Softgoal	Viable Healthcare Service
Clarity	2 - Softgoal	Viable Healthcare Service
Completeness	2 - Softgoal	Viable Healthcare Service
Accuracy	2 - Softgoal	Accountable
Clarity	2 - Softgoal	Accountable
Completeness	2 - Softgoal	Accountable
Accuracy	2 - Softgoal	Thustworthy [Healthcare System]
Clarity	2 - Softgoal	Thustworthy [Healthcare System]
Completeness	2 - Softgoal	Thustworthy [Healthcare System]

Atributo de transparência	Regra de produção	Elementos do iStar
Accuracy	2 - Softgoal	Flexible Treatment Plan
Clarity	2 - Softgoal	Flexible Treatment Plan
Completeness	2 - Softgoal	Flexible Treatment Plan
Accuracy	2 - Softgoal	Adhere To Treatment Plan
Clarity	2 - Softgoal	Adhere To Treatment Plan
Completeness	2 - Softgoal	Adhere To Treatment Plan
Accuracy	2 - Softgoal	Legitimate User Only
Clarity	2 - Softgoal	Legitimate User Only
Completeness	2 - Softgoal	Legitimate User Only
Accuracy	2 - Softgoal	Accommodating Daily Adjustments
Clarity	2 - Softgoal	Accommodating Daily Adjustments
Completeness	2 - Softgoal	Accommodating Daily Adjustments
Accuracy	2 - Softgoal	Confidencial Medical Data
Clarity	2 - Softgoal	Confidencial Medical Data
Completeness	2 - Softgoal	Confidencial Medical Data
Accuracy	2 - Softgoal	Min. Intrusion
Clarity	2 - Softgoal	Min. Intrusion
Completeness	2 - Softgoal	Min. Intrusion
Accuracy	2 - Softgoal	Time Saving
Clarity	2 - Softgoal	Time Saving
Completeness	2 - Softgoal	Time Saving
Accuracy	2 - Softgoal	Quality of Care
Clarity	2 - Softgoal	Quality of Care

Atributo de transparência	Regra de produção	Elementos do iStar
Completeness	2 - Softgoal	Quality of Care
Accuracy	2 - Softgoal	Normal Lifestyle
Clarity	2 - Softgoal	Normal Lifestyle
Completeness	2 - Softgoal	Normal Lifestyle
Accuracy	2 - Softgoal	Privacy
Clarity	2 - Softgoal	Privacy
Completeness	2 - Softgoal	Privacy
Divisibility	3 - Intentionality detailed (task decomposition)	Patient Centred Treatment <-> Execute Treatment Plan
Compositionality	3 - Intentionality detailed (task decomposition)	Patient Centred Treatment <-> Execute Treatment Plan
Divisibility	3 - Intentionality detailed (task decomposition)	Patient Centred Treatment <-> Design Treatment Plan
Compositionality	3 - Intentionality detailed (task decomposition)	Patient Centred Treatment <-> Design Treatment Plan
Divisibility	3 - Intentionality detailed (task decomposition)	Patient Centred Care <-> Plan Life Activities
Compositionality	3 - Intentionality detailed (task decomposition)	Patient Centred Care <-> Plan Life Activities
Divisibility	3 - Intentionality detailed (task decomposition)	Patient Centred Care <-> Flow Customized Treatment Plan
Compositionality	3 - Intentionality detailed (task decomposition)	Patient Centred Care <-> Flow Customized Treatment Plan
Validity	4 - Alternative operationalization (means-end link)	Keep Well <-> Provider Centred Care
Detailing	4 - Alternative operationalization (means-end link)	Keep Well <-> Provider Centred Care
Integrity	4 - Alternative operationalization (means-end link)	Keep Well <-> Provider Centred Care
Validity	4 - Alternative operationalization (means-end link)	Keep Well <-> Patient Centred Care
Detailing	4 - Alternative operationalization (means-end link)	Keep Well <-> Patient Centred Care
Integrity	4 - Alternative operationalization (means-end link)	Keep Well <-> Patient Centred Care
Operability	5.1 - Strategic dependence of resource	Controlling Access <-> Proof Identity <-> Obtain Patient Info
Operability	5.1 - Strategic dependence of resource	Analizing Patient Reports <-> Patient Condition Reports <-> Monitoring

Atributo de transparência	Regra de produção	Elementos do iStar
Traceability	5.2 - Strategic dependence of goal	Obtain Patient Info <-> Access Ganted [Request] <-> Controlling Access
Verifiability	5.2 - Strategic dependence of goal	Obtain Patient Info <-> Access Ganted [Request] <-> Controlling Access
Clarity	5.2 - Strategic dependence of goal	Obtain Patient Info <-> Access Ganted [Request] <-> Controlling Access
Compositionality	5.3 - Strategic dependence of softgoal	Regulator <-> Accountable <-> Healthcare Provider
Performance	8 - Responsibility(role)	HealthCare Provider Software Agent <-> Analizing Patient Reports
Divisibility	8 - Responsibility(role)	HealthCare Provider Software Agent <-> Analizing Patient Reports
Performance	8 - Responsibility(role)	HealthCare Provider Software Agent <-> Obtain Patient Info
Divisibility	8 - Responsibility(role)	HealthCare Provider Software Agent <-> Obtain Patient Info
Performance	8 - Responsibility(role)	Patient Assistant Software Agent <-> Monitoring
Divisibility	8 - Responsibility(role)	Patient Assistant Software Agent <-> Monitoring
Performance	8 - Responsibility(role)	Patient Assistant Software Agent <-> Controlling Access
Divisibility	8 - Responsibility(role)	Patient Assistant Software Agent <-> Controlling Access
Traceability	9 - Contribution +-	Patient Centred Treatment <-> Effective Treatment
Dependence	9 - Contribution +-	Patient Centred Treatment <-> Effective Treatment
Clarity	9 - Contribution +-	Patient Centred Treatment <-> Effective Treatment
Traceability	9 - Contribution +-	Patient Centred Treatment <-> Efficient Operations
Dependence	9 - Contribution +-	Patient Centred Treatment <-> Efficient Operations
Clarity	9 - Contribution +-	Patient Centred Treatment <-> Efficient Operations
Traceability	9 - Contribution +-	Effective Treatment <-> Viable Healthcare Service
Dependence	9 - Contribution +-	Effective Treatment <-> Viable Healthcare Service
Clarity	9 - Contribution +-	Effective Treatment <-> Viable Healthcare Service
Traceability	9 - Contribution +-	Efficient Operations <-> Viable Healthcare Service
Dependence	9 - Contribution +-	Efficient Operations <-> Viable Healthcare Service
Clarity	9 - Contribution +-	Efficient Operations <-> Viable Healthcare Service
Traceability	9 - Contribution +-	Restrict Access <-> Confidencial Medical Data

Atributo de transparência	Regra de produção	Elementos do iStar
Dependence	9 - Contribution +-	Restrict Access <-> Confidencial Medical Data
Clarity	9 - Contribution +-	Restrict Access <-> Confidencial Medical Data
Traceability	9 - Contribution +-	Confidencial Medical Data <-> Privacy
Dependence	9 - Contribution +-	Confidencial Medical Data <-> Privacy
Clarity	9 - Contribution +-	Confidencial Medical Data <-> Privacy
Traceability	9 - Contribution +-	Min. Intrusion <-> Privacy
Dependence	9 - Contribution +-	Min. Intrusion <-> Privacy
Clarity	9 - Contribution +-	Min. Intrusion <-> Privacy
Traceability	9 - Contribution +-	Patient Centred Care <-> Min. Intrusion
Dependence	9 - Contribution +-	Patient Centred Care <-> Min. Intrusion
Clarity	9 - Contribution +-	Patient Centred Care <-> Min. Intrusion
Traceability	9 - Contribution +-	Patient Centred Care <-> Time Saving
Dependence	9 - Contribution +-	Patient Centred Care <-> Time Saving
Clarity	9 - Contribution +-	Patient Centred Care <-> Time Saving
Traceability	9 - Contribution +-	Provider Centred Care <-> Quality of Care
Dependence	9 - Contribution +-	Provider Centred Care <-> Quality of Care
Clarity	9 - Contribution +-	Provider Centred Care <-> Quality of Care
Traceability	9 - Contribution +-	Patient Centred Care <-> Quality of Care
Dependence	9 - Contribution +-	Patient Centred Care <-> Quality of Care
Clarity	9 - Contribution +-	Patient Centred Care <-> Quality of Care
Traceability	9 - Contribution +-	Provider Centred Care <-> Normal Lifestyle
Dependence	9 - Contribution +-	Provider Centred Care <-> Normal Lifestyle
Clarity	9 - Contribution +-	Provider Centred Care <-> Normal Lifestyle
Traceability	9 - Contribution +-	Patient Centred Care <-> Normal Lifestyle
Dependence	9 - Contribution +-	Patient Centred Care <-> Normal Lifestyle

Clarity	9 - Contribution +-	Patient Centred Care <-> Normal Lifestyle
Traceability	9 - Contribution +-	Time Saving <-> Normal Lifestyle
Dependence	9 - Contribution +-	Time Saving <-> Normal Lifestyle
Clarity	9 - Contribution +-	Time Saving <-> Normal Lifestyle

APÊNDICE 5: Atributos de Transparência Ausentes no Sistema *Patient-Centred Care*

Atributo de transparência	**Regra de produção**	**Elementos do iStar**
Detail	Not alternative of operationalization	Conditions monitored
Validity	Not alternative of operationalization	Conditions monitored
Integrity	Not alternative of operationalization	Conditions monitored
Detail	Not alternative of operationalization	Access Ganted [Request]
Validity	Not alternative of operationalization	Access Ganted [Request]
Integrity	Not alternative of operationalization	Access Ganted [Request]
Detail	Not alternative of operationalization	Vital Signs Monitored
Validity	Not alternative of operationalization	Vital Signs Monitored
Integrity	Not alternative of operationalization	Vital Signs Monitored
Traceability	Not Contribution +-	Accountable
Dependence	Not Contribution +-	Accountable
Clarity	Not Contribution +-	Accountable
Traceability	Not Contribution +-	Thustworthy [Healthcare System]
Dependence	Not Contribution +-	Thustworthy [Healthcare System]
Clarity	Not Contribution +-	Thustworthy [Healthcare System]
Traceability	Not Contribution +-	Flexible Treatment Plan
Dependence	Not Contribution +-	Flexible Treatment Plan
Clarity	Not Contribution +-	Flexible Treatment Plan
Traceability	Not Contribution +-	Adhere To Treatment Plan
Dependence	Not Contribution +-	Adhere To Treatment Plan
Clarity	Not Contribution +-	Adhere To Treatment Plan
Traceability	Not Contribution +-	Legitimate User Only
Dependence	Not Contribution +-	Legitimate User Only

Atributo de transparência	Regra de produção	Elementos do iStar
Clarity	Not Contribution +-	Legitimate User Only
Traceability	Not Contribution +-	Accommodating Daily Adjustments
Dependence	Not Contribution +-	Accommodating Daily Adjustments
Clarity	Not Contribution +-	Accommodating Daily Adjustments
Traceability	Not dependence of goal	HealthCare Provider Software Agent
Clarity	Not dependence of goal	HealthCare Provider Software Agent
Traceability	Not dependence of goal	Monitoring
Clarity	Not dependence of goal	Monitoring
Traceability	Not dependence of goal	Patient Assistant Software Agent
Clarity	Not dependence of goal	Patient Assistant Software Agent
Traceability	Not dependence of goal	Healthcare Provider
Clarity	Not dependence of goal	Healthcare Provider
Traceability	Not dependence of goal	Patient
Clarity	Not dependence of goal	Patient
Verifiability	Not dependence of goal	HealthCare Provider Software Agent
Verifiability	Not dependence of goal	Monitoring
Verifiability	Not dependence of goal	Patient Assistant Software Agent
Verifiability	Not dependence of goal	Healthcare Provider
Verifiability	Not dependence of goal	Patient
Compositionality	Not dependence of softgoal	HealthCare Provider Software Agent
Compositionality	Not dependence of softgoal	Monitoring
Compositionality	Not dependence of softgoal	Patient Assistant Software Agent
Compositionality	Not dependence of softgoal	Healthcare Provider

Atributo de transparência	Regra de produção	Elementos do iStar
Compositionality	Not dependence of softgoal	Patient
Operability	Not dependence resource	HealthCare Provider Software Agent
Operability	Not dependence resource	Monitoring
Operability	Not dependence resource	Patient Assistant Software Agent
Operability	Not dependence resource	Healthcare Provider
Operability	Not dependence resource	Patient
Compositionality	Not detailed intention	Execute Treatment Plan
Divisibility	Not detailed intention	Execute Treatment Plan
Compositionality	Not detailed intention	Design Treatment Plan
Divisibility	Not detailed intention	Design Treatment Plan
Compositionality	Not detailed intention	Customize Treatment Plan
Divisibility	Not detailed intention	Customize Treatment Plan
Compositionality	Not detailed intention	Plan Life Activities
Divisibility	Not detailed intention	Plan Life Activities
Compositionality	Not detailed intention	Flow Customized Treatment Plan
Divisibility	Not detailed intention	Flow Customized Treatment Plan
Compositionality	Not detailed intention	Restrict Access
Divisibility	Not detailed intention	Restrict Access
Compositionality	Not detailed intention	Provider Centred Care
Divisibility	Not detailed intention	Provider Centred Care
Traceability	Not intentionality	Analizing Patient Reports
Verifiability	Not intentionality	Analizing Patient Reports
Traceability	Not intentionality	Obtain Patient Info

Atributo de transparência	Regra de produção	Elementos do iStar
Verifiability	Not intentionality	Obtain Patient Info
Traceability	Not intentionality	Patient Assistant Software Agent
Verifiability	Not intentionality	Patient Assistant Software Agent
Traceability	Not intentionality	Healthcare Provider
Verifiability	Not intentionality	Healthcare Provider
Traceability	Not intentionality	Regulator
Verifiability	Not intentionality	Regulator
Verifiability	Not organizational structure (position)	HealthCare Provider Software Agent
Verifiability	Not organizational structure (position)	Analizing Patient Reports
Verifiability	Not organizational structure (position)	Obtain Patient Info
Verifiability	Not organizational structure (position)	Monitoring
Verifiability	Not organizational structure (position)	Controlling Access
Verifiability	Not organizational structure (position)	Patient Assistant Software Agent
Verifiability	Not organizational structure (position)	Healthcare Provider
Verifiability	Not organizational structure (position)	Regulator
Verifiability	Not organizational structure (position)	Patient
Performance	Not Responsibility (role)	Analizing Patient Reports
Divisibility	Not Responsibility (role)	Analizing Patient Reports
Performance	Not Responsibility (role)	Obtain Patient Info
Divisibility	Not Responsibility (role)	Obtain Patient Info
Performance	Not Responsibility (role)	Monitoring
Divisibility	Not Responsibility (role)	Monitoring
Performance	Not Responsibility (role)	Controlling Access

Atributo de transparência	Regra de produção	Elementos do iStar
Divisibility	Not Responsibility (role)	Controlling Access
Performance	Not Responsibility (role)	Healthcare Provider
Divisibility	Not Responsibility (role)	Healthcare Provider
Performance	Not Responsibility (role)	Regulator
Divisibility	Not Responsibility (role)	Regulator
Performance	Not Responsibility (role)	Patient
Divisibility	Not Responsibility (role)	Patient
Availability	Not Taxonomy of actor	HealthCare Provider Software Agent
Availability	Not Taxonomy of actor	Analizing Patient Reports
Availability	Not Taxonomy of actor	Obtain Patient Info
Availability	Not Taxonomy of actor	Monitoring
Availability	Not Taxonomy of actor	Controlling Access
Availability	Not Taxonomy of actor	Patient Assistant Software Agent
Availability	Not Taxonomy of actor	Healthcare Provider
Availability	Not Taxonomy of actor	Regulator
Availability	Not Taxonomy of actor	Patient

APÊNDICE 6: Especificação das Regras de Produção

Nessa seção é abordada a lógica utilizada para implementar as regras de produção. O Código 1 mostra a implementação da regra de produção *ruleIntentionality* para verificar a presença do atributo de transparência *intencionalidade*. Nela há duas condições necessárias para que a ação seja disparada, linhas 2 e 3. Se na base de fatos houver um elemento do tipo *actor,* outro do tipo *goal* e o *idActor* do actor for igual ao *idActor* do elemento *goal*, então ocorrerá o disparo dessa regra. Isso significa que para toda meta na base de dados ela será relacionada ao seu respectivo ator indicando assim a intencionalidade do mesmo. Quando a regra de intencionalidade é disparada dois novos fatos são adicionados à base de dados, o fato *attribute* indicando a presença do atributo *verificabilidade* e outro para indicar a presença do atributo *rastreabilidade*, conforme linhas 5 à 19.

```
1   (defrule ruleIntentionality "1 - Intentionality (actor e goal)"
2           (element (id ?idGoal)  (name ?nameGoal) (type "goal") (idActor ?idActor ) )
3           (element (id ?idActor) (name ?nameActor)                                    )
4           =>
5           (assert (attribute
6                                   (attribute "Traceability")
7                                   (rule   "1 - Intentionality (actor e goal)" )
8                                   (feature "actor and goal")
9                                   (detail (sym-cat ?nameActor " <-> " ?nameGoal))
10
11                          )
12          )
13          (assert (attribute
14                                   (attribute "Verifiability")
15                                   (rule   "1 - Intentionality (actor e goal)" )
16                                   (feature "actor and goal")
17                                   (detail (sym-cat ?nameActor " <-> " ?nameGoal))
18                  )
19          )
20  )
21
```

Código 1: A regra ruleIntentionality.

Quando um atributo é adicionado à base de fatos do sistema de verificação, é armazenada também a regra responsável por sua identificação, o conceito do *framework* utilizado na regra e também os elementos do modelo de requisitos que disparam essa regra. Permitindo assim rastrear as informações que originaram a

identificação de um dado atributo na especificação de requisitos em questão. Observe esse fato nas linhas 7 à 9 e 14 à 17, do Código 1.

O iStar permite representar metas flexíveis através do elemento *softgoal,* indicando que os critérios para a satisfação do objetivo dependem do ponto de vista do ator. Segundo Oliveira et al. (2007), as metas flexíveis permitem identificar os seguintes atributos de transparência: *clareza*, *acurácia* e *completude*. Nesse sentido, a regra *ruleSoftgoal* foi implementada conforme mostra o Código 2. A condição necessária para que a regra seja disparada é a presença de um elemento do tipo *softgoal* na base de fatos. Quando ocorre essa correspondência na base de dados, três novos fatos *attribute* são criados, os atributos de transparência *clareza*, *acurácia* e *completude*.

```
1   (defrule ruleSoftgoal "2 - Softgoal"
2           (element (id ?idSoftgoal)  (name ?nameSoftgoal) (type "softgoal") (idActor ?idActor ) )
3           =>
4           (assert (attribute
5                                   (attribute "Completeness")
6                                   (rule   "2 - Softgoal")
7                                   (feature "softgoal")
8                                   (detail ?nameSoftgoal)
9                   )
10          )
11          (assert (attribute
12                                  (attribute "Clarity")
13                                  (rule   "2 - Softgoal")
14                                  (feature "softgoal")
15                                  (detail ?nameSoftgoal)
16                  )
17          )
18          (assert (attribute
19                                  (attribute "Accuracy")
20                                  (rule   "2 - Softgoal")
21                                  (feature "softgoal")
22                                  (detail ?nameSoftgoal)
23                  )
24          )
25  )
26
```

Código 2: A regra ruleSoftgoal.

Conforme mostra a Tabela 9, os atributos de transparência *compositividade* e *divisibilidade* são identificados pela regra *ruleIntentionalityDetailed*. Quando o

modelo possui decomposição de tarefas esses atributos são identificados. O Código 3 mostra a implementação dessa regra.

```
1  (defrule ruleIntentionalityDetailed "3 - Intentionality detailed (task decomposition)"
2    (element        (id ?idTask)          (name ?nameTask)   (type "task")                                  )
3    (elementLink    (label ?label)     (type "decomposition")  (source ?source) (target ?idTask)  )
4    (element        (id ?source)          (name ?nameElement)                                                )
5          =>
6          (assert (attribute
7                                        (attribute "Compositionality")
8                                        (rule   "3 - Intentionality detailed (task decomposition)")
9                                        (feature "decomposition link")
10                                       (detail (sym-cat ?nameTask " <-> " ?nameElement))
11                 )
12         )
13         (assert (attribute
14                                       (attribute "Divisibility")
15                                       (rule   "3 - Intentionality detailed (task decomposition)")
16                                       (feature "decomposition link")
17                                       (detail (sym-cat ?nameTask " <-> " ?nameElement))
18                 )
19         )
20 )
21
```

Código 3: A regra ruleIntentionalityDetailed.

Três condições são testadas na seção antecedente dessa regra, conforme linhas 2, 3 e 4 do Código 3. A presença de um elemento do tipo *task,* um conector do tipo *decomposition* e o elemento relacionado à decomposição. Ou seja, se na base de fatos houver uma tarefa que foi decomposta em outro(s) elemento(s) então a regra será disparada e dois novos fatos farão parte da base de dados, linhas 6 à 19. Observe a correspondência entre o atributo (*id* ***?idTask****)* no primeiro elemento, linha 2, com o segundo elemento nos atributos (*target* ***?idTask***) e (*source* ***?source****)*, linha 3. Com o terceiro elemento, linha 4, atributo (*source* ***?source****)*.

A regra *ruleAlternativeOfOperationalization* permite identificar as alternativas de operacionalização de metas, indicando assim a presença dos atributos de transparência *integridade*, *detalhamento* e *validade*. O Código 4 exibe a lógica empregada para obter esse resultado. Ela analisa as alternativas disponíveis para que as metas sejam alcançadas. Os elementos do tipo *goal* e os conectores *means-end* são utilizados para identificar quais são os elementos que são alternativas de operacionalização para as metas representadas no modelo.

```
1  (defrule ruleAlternativeOfOperationalization "4 - Alternative of operationalization "
2    (element    (id ?idGoal)   (name ?nameGoal)   (type "goal")     (idActor ?idActor ) )
3    (element    (id ?idElement) (name ?nameElement) (type ?typeElement)(idActor ?idActor ) )
4    (elementLink (type "means-end")  (source ?idElement)   (target ?idGoal)              )
5          =>
6          (assert (attribute
7                                (attribute "Integrity")
8                                (rule   "4 - Alternative operationalization (means-end link)")
9                                (feature "means-end link")
10                               (detail (sym-cat ?nameGoal " <-> " ?nameElement))
11                 )
12         )
13         (assert (attribute
14                               (attribute "Detailing")
15                               (rule   "4 - Alternative operationalization (means-end link)")
16                               (feature "means-end link")
17                               (detail (sym-cat ?nameGoal " <-> " ?nameElement))
18                 )
19         )
20         (assert (attribute
21                               (attribute "Validity")
22                               (rule   "4 - Alternative operationalization (means-end link)")
23                               (feature "means-end link")
24                               (detail (sym-cat ?nameGoal " <-> " ?nameElement))
25                 )
26         )
27
28 )
29
```

Código 4: A regra ruleAlternativeOfOperationalization.

Na parte antecedente da regra *ruleAlternativeOfOperationalization* é testado se o elemento *goal* possui uma alternativa de operacionalização através do fato *elementLink* relacionando o elemento que operacionaliza essa meta.

Conforme relacionado na Tabela 9, quando essa regra é disparada os atributos *integridade*, *detalhamento* e *validade* são criados na base de fatos.

A dependência estratégica de recursos permite identificar o atributo *operabilidade*. A regra *ruleStrategicDependenceOfResource* verifica no modelo se há a ocorrência dessa situação. Quando o analista modela a dependência entre os atores em relação aos recursos essa regra cria um novo fato na base de dados referente ao atributo *operabilidade*. O motor de inferência verifica se há o relacionamento de dependência entre dois atores e se essa relação é de dependência de recursos. Observa-se no Código 5 que o teste utiliza dois fatos *elementLink* e três fatos *element*. O *elementLink* representa a conexão entre o

depender, o *dependum* e o *dependee*. Os fatos *element* representam o *depender*, *dependum* e o *dependee* respectivamente.

```
1  (defrule ruleStrategicDependenceOfResource "5.1 - Strategic dependence of resource"
2          (elementLink    (type "dependence")  (source ?idDepender) (target ?idElement)   )
3          (element        (id ?idDepender)     (name ?nameDepender) (type "actor")       )
4          (element        (id ?idElement)      (name ?nameElement)  (type "resource")
5                          (idActor ?idActor ) )
6          (elementLink    (type "dependence")  (source ?idElement)  (target ?idDependee)  )
7          (element         (id ?idDependee)    (name ?nameDependee) (type "actor")       )
8          =>
9          (assert (attribute
10                                        (attribute "Operability")
11                                        (rule   "5.1 - Strategic dependence of resource")
12                                        (feature "dependence link, resource")
13                                        (detail (sym-cat ?nameDepender " <->
14                                                 " ?nameElement " <-> " ?nameDependee))
15                          )
16         )
17
18 )
19
```

Código 5: A regra ruleStrategicDependenceOfResource.

A dependência estratégica de *goal* permite identificar os atributos *clareza, verificabilidade e rastreabilidade*. A regra *ruleStrategicDependenceOfGoal* verifica se no modelo ocorre esse fato. Se no modelo de requisitos há a dependência entre os atores em relação a metas, essa regra cria três novos fatos na base de dados referente aos atributos *clareza, verificabilidade e rastreabilidade*. Essa regra testa se há o relacionamento de dependência entre dois atores e se essa relação é de dependência de meta. O Código 6 apresenta dois fatos *elementLink* e três fatos *element* que formam a primeira parte da regra. Os fatos *element* representam o *depender*, *dependum* e o *dependee* respectivamente, enquanto o *elementLink* representa a conexão entre o *depender*, o *dependum* e o *dependee*. Nesse caso o tipo do elemento *dependum* é *goal.*

```
1  (defrule ruleStrategicDependenceOfGoal "5.2 - Strategic dependence of goal"
2          (elementLink    (type "dependence")  (source ?idDepender) (target ?idElement)  )
3          (element        (id ?idDepender)     (name ?nameDepender) (type "actor")           )
4          (element        (id ?idElement)      (name ?nameElement)  (type "goal")
5                          (idActor ?idActor ) )
6          (elementLink    (type "dependence")  (source ?idElement)  (target ?idDependee)   )
7          (element        (id ?idDependee)     (name ?nameDependee) (type "actor")           )
8          =>
9          (assert (attribute
10                                      (attribute "Clarity")
11                                      (rule   "5.2 - Strategic dependence of goal")
12                                      (feature "dependence link, goal")
13                                       (detail (sym-cat ?nameDepender " <-> "
14                                      ?nameElement " <-> " ?nameDependee))
15                          )
16         )
17         (assert (attribute
18                                      (attribute "Verifiability")
19                                      (rule   "5.2 - Strategic dependence of goal")
20                                      (feature "dependence link, goal")
21                                       (detail (sym-cat ?nameDepender " <-> "
22                                      ?nameElement " <-> " ?nameDependee))
23                          )
24         )
25         (assert (attribute
26                                      (attribute "Traceability")
27                                      (rule   "5.2 - Strategic dependence of goal")
28                                      (feature "dependence link, goal")
29                                       (detail (sym-cat ?nameDepender " <-> "
30                                      ?nameElement " <-> " ?nameDependee))
31                          )
32         )
33
34 )
```

Código 6: A regra ruleStrategicDependenceOfGoal.

A regra *ruleStrategicDependenceOfSofgoal* foi projetada para identificar a presença de dependência de *softgoal* entre os atores representados no modelo de requisitos. Ela testa se há o relacionamento de dependência entre dois atores e se essa relação é de dependência de *softgoal*.

O Código 7 mostra dois fatos *elementLink* e três fatos *element* que formam a primeira parte da regra. Os fatos *element* representam o *depender*, *dependum* e o *dependee* respectivamente, enquanto o *elementLink* representa a conexão entre o *depender*, o *dependum* e o *dependee*. Nessa regra o tipo do elemento *dependum* é soft*goal.*

```
1  (defrule ruleStrategicDependenceOfSoftgoal "5.3 - Strategic dependence of softgoal"
2          (elementLink    (type "dependence")  (source ?idDepender) (target ?idElement) )
3          (element        (id ?idDepender)     (name ?nameDepender) (type "actor")     )
4          (element        (id ?idElement)      (name ?nameElement)  (type "softgoal")
5                                                                    (idActor ?idActor ) )
6          (elementLink    (type "dependence")  (source ?idElement)  (target ?idDependee)  )
7          (element        (id ?idDependee)     (name ?nameDependee) (type "actor")  )
8          =>
9          (assert (attribute
10                                      (attribute "Compositionality")
11                                      (rule   "5.3 - Strategic dependence of softgoal")
12                                      (feature "dependence link, softgoal")
13                                      (detail (sym-cat ?nameDepender " <-> "
14                                      ?nameElement " <-> "
15                                      ?nameDependee))
16                          )
17         )
18
19 )
20
21
```

Código 7: A regra ruleStrategicDependenceOfSoftgoal.

Quando as condições estabelecidas na parte antecedente dessa regra forem satisfeitas, será criado um novo fato na base de dados do sistema de verificação informando a presença do atributo de transparência *compositividade*.

A classificação dos atores representados no modelo de requisitos permite identificar a presença do atributo de transparência *disponibilidade* conforme apresentado na Tabela 9. Nesse sentido a regra *ruleTaxonomyOfActors* foi implementada conforme mostra o Código 8.

Na linha 2 foi declarado o fato *elementLink,* a partir dos atributos *source* e *target* é possível relacionar os atores envolvidos na classificação. Quando houver na base de fatos um conector *is_a* ou *INS* entre dois atores, então a regra criará um novo fato *attribute* na base de dados indicando a presença do atributo de transparência *disponibilidade*.

```
(defrule ruleTaxonomyOfActors "6 - Taxonomy of actors (actors link INS e ISA)"
  (elementLink (label "is_a" | "INS")  (source ?sourceActorLink)  (target ?targetActorLink) )
  (element     (id ?sourceActorLink)  (name ?nameActor))
  ( element     (id ?targetActorLink)  (name ?nameAgent)  )
        =>
        (assert (attribute
                                (attribute "Availability")
                                (rule   "6 - Taxonomy of actors (actors link INS e ISA)" )
                                (feature "actors link, actor, agent")
                                (detail (sym-cat ?nameActor " <-> "?nameAgent ))
                        )
        )
)

```

Código 8: A regra ruleTaxonomyOfActors.

A regra *ruleOrganizationalStructure*, apresentada no Código 9, verifica no modelo de requisitos a presença do atributo de transparência *verificabilidade*. A partir do fato *elementLink* do tipo *occupies* ela relaciona a posição ocupada por um dado ator.

```

(defrule ruleOrganizationalStructure "7 - The organizational structure (position)"
        (elementLink  (label "occupies")  (type ?typeActorLink)
                        (source ?sourceActorLink)  (target ?targetActorLink)  )
        ( element     (id ?sourceActorLink)    (name ?nameActor)    )
        ( element     (id ?targetActorLink)    (name ?nameElementPosition)  )
        =>
        (assert (attribute
                                        (attribute "Verifiability")
                                        (rule   "7 - The organizational structure (position)")
                                        (feature "actors link (occupies)")
                                         (detail (sym-cat ?nameActor
                                        " <-> " ?nameElementPosition))
                        )
        )
)

```

Código 9: A regra ruleOrganizationalStructure.

A lógica utilizada na implementação da regra *ruleResponsibility* é semelhante a empregrada na regra *ruleOrganizationalStructure*, apresentada no Código 9. A partir do fato *elementLink* do tipo *plays* ela relaciona o papel desempenhado por um dado ator. O Código 10 exibe a estrutura da regra *ruleResponsibility,* quando houver a relação de *plays* na base de fatos a regra vai adicionar à base de dados os atributos de transparência *divisibilidade* e *desempenho*.

```
1
2   (defrule ruleResponsibility "8 - Responsibility(role)"
3           (elementLink (label "plays")  (type ?typeActorLink)
4                       (source ?sourceActorLink)  (target ?targetActorLink)  )
5           (element    (id ?sourceActorLink)    (name ?nameActor)      )
6           (element    (id ?targetActorLink)    (name ?nameElementRole)   )
7           =>
8           (assert (attribute
9                                       (attribute "Divisibility")
10                                      (rule  "8 - Responsibility(role)")
11                                      (feature "actors link (plays)")
12                                      (detail (sym-cat  ?nameActor " <-> "
13                                      ?nameElementRole))
14                      )
15          )
16          (assert (attribute
17                                      (attribute "Performance")
18                                      (rule  "8 - Responsibility(role)")
19                                      (feature "actors link (plays)")
20                                       (detail (sym-cat  ?nameActor " <-> "
21                                      ?nameElementRole))
22                      )
23          )
24  )
```

Código 10: A regra ruleResponsibility.

As contribuições positivas e negativas representadas com o framework iStar permitem mostrar as contribuições que os elementos intencionais *tarefa*, *meta*, *meta flexível*, e *crença* podem fornecer aos *softgoals.* Nesse sentido a regra *ruleContribution* foi projetada conforme mostra o Código 11. A lógica usada consiste em identificar o relacionamento entre um elemento e a meta flexível através do fato *elementLink* do tipo *contribution.* Quando essa condição for verdadeira a regra adicionará três novos fatos à base de dados contendo os atributos de transparência *clareza, dependência e rastreabilidade*.

```
1   (defrule ruleContribution "9 - Contribution +-"
2           (elementLink (label ?label) (type "contribution")
3                       (source ?source) (target ?target)  )
4           (element    (id ?source)   (name ?nameElement)  )
5           (element    (id ?target)   (name ?nameSoftgoal )   )
6           =>
7           (assert (attribute
8                                       (attribute "Clarity")
9                                       (rule   "9 - Contribution +-")
10                                      (feature "contribution +-")
```

```
11                                  (detail (sym-cat ?nameElement " <-> "
12                                  ?nameSoftgoal))
13                          )
14          )
15          (assert (attribute
16                                  (attribute "Dependence")
17                                  (rule   "9 - Contribution +-")
18                                  (feature "contribution +-")
19                                   (detail (sym-cat ?nameElement " <-> "
20                                   ?nameSoftgoal))
21                          )
22          )
23          (assert (attribute
24                                  (attribute "Traceability")
25                                  (rule   "9 - Contribution +-")
26                                  (feature "contribution +-")
27                                  (detail (sym-cat ?nameElement " <-> "
28                                   ?nameSoftgoal))
29                          )
30          )
31  )
```

Código 11: A regra ruleContribution.

A regra ruleDenoteSynonym verifica se no modelo de requisitos representado com o *framework* iStar foram mencionados termos que são sinônimos dos atributos de transparência. Essa abordagem visa obter a relação semântica entre o modelo de requisitos e os atributos de transparência.

O Código 12 apresenta a estrutura dessa regra. A partir de um elemento do modelo, fato *element,* busca-se sua correspondência com a base de fatos contendo os sinônimos dos atributos, linhas 2 e 3. Com o sinônimo identificado, obtém-se o referido atributo de transparência, linha 4. Quando essas três condições são verdadeiras, a regra é disparada mostrando a relação entre o termo informado e o atributo de transparência.

```
1   (defrule ruleDenoteSynonym
2           (element (id ?id) (name ?name) (type ?type) )
3           (word (id ?idAttribute) (word ?nameAttribute) )
4           (word (id ?idSim) (word ?name) (idAttribute ?idAttribute))
5           =>
6           (printout t ?name " denote " ?nameAttribute crlf)
7   )
8
```

Código 12: A regra ruleDenoteSynonym.

A regra *ruleNotIntentionality,* Código 13, identifica no modelo a falta de intencionalidade. A partir do fato *element* do tipo *actor* é verificado se um ator não possui relação com um elemento do tipo *goal*. Ou seja, para haver intencionalidade o objetivo do ator deve ser representado. Na linha 3 o comando *not* é utilizado para identificar um ator sem meta.

```
1  (defrule ruleNotIntentionality "Indicates the lack intentionality"
2     (element  (id ?idActor)  (name ?nameActor) (type "actor") )
3     (not (element (id ?idGoal)  (name ?nameGoal) (type "goal") (idActor ?idActor )  )  )
4           =>
5           (assert (attribute
6                                 (attribute "Verifiability")
7                                 (rule    "Not intentionality")
8                                 (feature "actor and goal")
9                                 (detail  ?nameActor)
10                          )
11          )
12          (assert (attribute
13                                (attribute "Traceability")
14                                (rule    "Not intentionality")
15                                (feature "actor and goal")
16                                (detail  ?nameActor)
17                          )
18          )
19 )
```

Código 13: A regra ruleNotIntentionality.

Quando a base de fatos contém um ator sem um relacionamento com um *goal,* são adicionados dois novos fatos *attribute* à base de dados. Um fato para indicar a falta de *verificabilidade* e outro para indicar a falta de *rastreabilidade*. Ou seja, sem intencionalidade no modelo ocorre a ausência desses atributos de transparência.

A representação de metas flexíveis permite identificar os atributos *completude*, *clareza* e *acurácia*. Quando o modelo não apresenta metas flexíveis, esses atributos não estarão no modelo de requisitos.

Conforme Código 14, quando a base de fatos não tiver o fato *element,* então serão acrescentados três novos fatos *attribute* indicando a ausência dos atributos *acurácia*, *clareza* e *completude*. Na linha 2 foi usado o comando *not* para verificar a falta desse elemento na base de fatos.

```
1
2   (defrule ruleNotSofgoal "Indicates the lack softgoal"
3           (not (element (id ?idGoal)  (name ?nameGoal) (type "softgoal") (idActor ?idActor ) ) )
4           =>
5           (assert (attribute
6                                   (attribute "Completeness")
7                                   (rule    "Not softgoal")
8                                   (feature "softgoal")
9                           )
10          )
11          (assert (attribute
12                              (attribute "Clarity")
13                                  (rule    "Not softgoal")
14                                  (feature "softgoal")
15                          )
16          )
17          (assert (attribute
18                                  (attribute "Accuracy")
19                                  (rule    "Not softgoal")
20                                  (feature "softgoal")
21                          )
22          )
23  )
```

Código 14: A regra ruleNotSofgoal

A regra *ruleNotDetailedIntention* verifica a falta de decomposição de tarefas no modelo de requisitos. Ou seja, a ausência de intencionalidade detalhada, conforme mostra a Tabela 9. A intencionalidade detalhada permite identificar os atributos *divisibilidade* e *compositividade*. Conforme mostra o Código 15, quando uma tarefa não possui um *elementLink* do tipo decomposition ela não sofre decomposição. Ou seja, essa regra identifica as tarefas que não sofreram decomposição, indicando assim a falta de intencionalidade detalhada.

```
1   (defrule ruleNotDetailedIntention "Indicates the lack of detailed intentionality"
2     (element (id ?idTask)  (name ?nameTask) (type "task") )
3     (not (elementLink (label ?label) (type "decomposition") (source ?source) (target ?idTask) ) )
4           =>
5           (assert (attribute
6                                   (attribute "Divisibility")
7                                   (rule    "Not detailed intention")
8                                   (feature "decomposition link")
9                                   (detail  ?nameTask)
10                          )
11          )
12          (assert (attribute
13                                  (attribute "Compositionality")
14                                  (rule    "Not detailed intention")
```

```
15                              (feature "decomposition link")
16                              (detail  ?nameTask)
17                      )
18          )
19  )
```

Código 15: A regra ruleNotDetailedIntention.

Essa regra é disparada quando a parte antecedente dessa regra é verdadeira, ou seja, na base de fatos há uma tarefa que não tem decomposição, criando assim dois novos fatos na base contendo os atributos *divisibilidade* e *compositividade*.

Essa regra verifica se há no modelo de requisitos um elemento do tipo *goal* sem relação com um link *means-end,* o que indica a falta de alternativa de operacionalização. Quando houver no modelo de requisitos uma meta sem alternativa de operacionalilzação, conforme declarado nas linhas 2 e 3 do Código 16, a regra adicionará à base de dados os fatos *attribute* indicando a falta dos atributos de transparência *integridade, validade* e *detalhamento.*

```
1   (defrule ruleNotAlternativeOfOperationalization "The lack of altern. of operationalization"
2      (element     (id ?idGoal)    (name ?nameGoal)    (type "goal") (idActor ?idActor ) )
3      (not (elementLink (type "means-end")  (source ?idElement)   (target ?idGoal) ) )
4           =>
5           (assert (attribute
6                                   (attribute "Integrity")
7                                   (rule   "Not alternative of operationalization")
8                                   (feature "means-end link")
9                                   (detail  ?nameGoal)
10                          )
11          )
12          (assert (attribute
13                                  (attribute "Validity")
14                                  (rule   "Not alternative of operationalization")
15                                  (feature "means-end link")
16                                  (detail  ?nameGoal)
17                          )
18          )
19          (assert (attribute
20                                  (attribute "Detail")
21                                  (rule   "Not alternative of operationalization")
22                                  (feature "means-end link")
23                                  (detail  ?nameGoal)
24                          )
25          )
26
27  )
28
```

Código 16: A regra ruleNotAlternativeOfOperationalization.

A regra *ruleNotDependence,* Código 17, verifica se no modelo de requisitos há elementos do tipo actor que não envolvidos em uma relação de dependencia estratégica. Para cada ator é identificada a relação com um *conector* do tipo *dependence*, quando essa relação for negativa a regra é disparada criando na base de fatos os fatos com o nome desse ator juntamente com os atributos que o relacionamento de dependência permite identificar.

```
1  (defrule ruleNotDependence "Indicates the lack of dependence"
2        (element  (id ?idDepender)  (name ?nameDepender) (type "actor") )
3        (not (elementLink   (type "dependence")  (source ?idDepender)
4                            (target ?idElement) ) )
5     =>
6        (assert (attribute
7                            (attribute "Operability")
8                            (rule   "Not dependence resource")
9                            (feature "dependence link")
10                           (detail  ?nameDepender)
11                   )
12       )
13       (assert (attribute
14                           (attribute "Clarity")
15                           (rule   "Not dependence of goal")
16                           (feature "dependence link")
17                           (detail  ?nameDepender)
18                   )
19       )
20       (assert (attribute
21                           (attribute "Verifiability")
22                           (rule   "Not dependence of goal ")
23                           (feature "dependence link")
24                           (detail  ?nameDepender)
25                   )
26       )
27       (assert (attribute
28                           (attribute "Traceability")
29                           (rule   "Not dependence of goal")
30                           (feature "dependence link")
31                           (detail  ?nameDepender)
32                   )       )
33       (assert (attribute
34                           (attribute "Compositionality")
35                           (rule   "Not dependence of softgoal")
36                           (feature "dependence link")
37                           (detail  ?nameDepender)
38                   )       )
```

```
39  )
40
```

Código 17: A regra ruleNotDependence.

A regra *ruleNotTaxonomyOfActors,* Código 18, identifica a ausência de classificação de atores no modelo de requisitos, quando houver um ator sem relacionamento com um conector *is_a* ou *INS* sua classificação não foi representada, indicando a falta do atributo *disponibilidade* nessa parte do modelo de requisitos. Quando ocorre essa situação, o fato attribute é adicionado à base de dados contendo o attributo *Availability*, a descrição da regra, o conceito do istar e o nome do ator.

```
1   (defrule ruleNotTaxonomyOfActors "Indicates the lack of taxonomy of actors"
2           (element (id ?sourceActorLink)  (name ?nameActor) (type "actor") )
3           (not (elementLink (label "is_a" | "INS")  (source ?sourceActorLink)  (target
4   ?targetActorLink)  )
5           =>
6           (assert (attribute
7                                   (attribute "Availability")
8                                   (rule   "Not Taxonomy of actor")
9                                   (feature "links is-a and INS")
10                                  (detail  ?nameActor)
11                          )
12          )
```

Código 18: A regra ruleNotTaxonomyOfActors.

A regra *ruleNotOrganizationalStructure* exibida no Código 19, verifica a ausência de informações sobre a estrutura organizacional na especificação de requisitos representada com o *Framework* iStar. O operador lógico *not* é usado para relacionar todos os atores para os quais não foram indicadas suas posições.

```
1   (defrule ruleNotOrganizationalStructure
2     (element (id ?sourceActorLink)  (name ?nameActor) (type "actor") )
3     (not (elementLink (label "occupies")  (type ?typeActorLink) (source
4   ?sourceActorLink)  (target ?targetActorLink) ) )
5           =>
6           (assert (attribute
7                                       (attribute "Verifiability")
8                                       (rule   "Not organizational structure (position)")
9                                       (feature "link (occupies)")
```

```
                                        (detail  ?nameActor)
                )       )       )
```

Código 19: A regra ruleNotOrganizationalStructure.

As informações sobre a estrutura organizacional permitem identificar o atributo de transparência *verificabilidade* conforme apresentado na Tabela 9, quando as condições da parte antecedente da regra *ruleNotOrganizationalStructure* são satisfeitas a regra dispara informando que o atributo *verificabilidade* não está presente nesse parte da especificação.

A descrição dos papéis desempenhados pelos atores no modelo de requisitos permite identificar os atributos *divisibilidade* e *desempenho*, conforme exibido na Tabela 9. Quando um ator não possuir essa informação tais atributos de transparência não serão identificados nessa parte do modelo. O Código 20 mostra a regra *ruleNotResponsibility* projetada para indicar essa ocorrência.

```

(defrule ruleNotResponsibility "Indicates the lack of responsibility of actors"
   (element (id ?sourceActorLink)  (name ?nameActor) (type "actor"))
   (not (elementLink (label "plays")  (type ?typeActorLink) (source ?sourceActorLink)
(target ?targetActorLink) ) )
=>
        (assert (attribute
                                        (attribute "Divisibility")
                                        (rule   "Not Responsibility (role)")
                                        (feature "link (plays)")
                                        (detail  ?nameActor)
                        )
        )
        (assert (attribute
                                        (attribute "Performance")
                                        (rule   "Not Responsibility (role)")
                                        (feature "link (plays)")
                                        (detail  ?nameActor)
                        )
        )

)
```

Código 20: A regra ruleNotResponsibility.

A regra *ruleNotContribution,* Código 21*,* aponta a ausência de conectores de contribuição nos elementos do tipo *softgoal.* Quando a especificação de requisitos apresenta um *softgoal* sem um *link contribution* a regra é disparada criando três

novos fatos na base de dados sinalizando a ausência dos atributos *clareza*, *dependência* e *rastreabilidade*.

```
1
2  (defrule ruleNotContribution "Indicates the lack of contributions +-"
3     (element (id ?target)  (name ?nameSoftgoal) (type "softgoal")  )
4     (not (elementLink (label ?label) (type "contribution")
5                   (source ?source) (target ?target) ) )
6           =>
7           (assert (attribute
8                                       (attribute "Clarity")
9                                       (rule   "Not Contribution +-")
10                                      (feature "contribution +-")
11                                      (detail  ?nameSoftgoal)
12                   )
13          )
14          (assert (attribute
15                                      (attribute "Dependence")
16                                      (rule   "Not Contribution +-")
17                                      (feature "contribution +-")
18                                      (detail  ?nameSoftgoal)
19                  )
20          )
21          (assert (attribute
22                                      (attribute "Traceability")
23                                      (rule   "Not Contribution +-")
24                                      (feature "contribution +-")
25                                      (detail  ?nameSoftgoal)
26                  )
27         )
28   )
```

Código 21: A regra ruleNotContribution.

A base de conhecimento desenvolvida nesse trabalho implementa as informações relacionadas na Tabela 4. Ou seja, a partir das características do *Framework* iStar que possibilitam a identificação de atributos de transparência foram implementadas as regras apresentadas nessa capítulo. Essas características foram obtidas a partir dos trabalhos de Leite e Cappelli, (2008) e Oliveira et al. (2007).

REFERÊNCIAS BIBLIOGRÁFICAS

BRASIL. **Código de Defesa do Consumidor**. Lei 8.078 de 11/09/90. Brasília, Diário Oficial da União, 1990.

BRASIL. Constituição (1988). **Constituição da República Federativa do Brasil**. Brasília, DF, Senado, 1998.

CAPPELLI, C. **Uma Abordagem para Transparência em Processos Organizacionais Utilizando Aspectos**. 2009. 328 f. Tese de Doutorado – Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro, RJ, 2009.

CAPPELLI, C.; LEITE, J. C. S. P. **Transparência de Processos Organizacionais.** In: Simpósio Internacional de Transparência nos Negócios, 2008, Rio de Janeiro, Brasil. **Anais**... Rio de Janeiro: Universidade Federal Fluminense, LATEC, 2008. Disponível em: <http://www.latec.uff.br/transparencia/anais.pt-br.php>. Acesso em: 25 jan. 2011.

CAPPELLI, C.; OLIVEIRA, A. PADUA; LEITE, J. **Exploring Business Process Transparency Concepts**. In: RE'07, 15th IEEE Joint International Requirements Engineering Conference, Delhi, India, 2007.

CARES, C.; FRANCH, X.; PERINI, A.; SUSI, A. **iStarML:** an XML-based Interchange Format for i* Models. In: Third International i* Workshop, 2008, Recife, Brasil. **Proceedings** ... Recife: Universidade Federal de Pernambuco, 2008. pp. 13-16. Disponível em <http://sunsite.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-322/>. Acesso em: 21 Jan. 2011.

DURAN, A.; BERNARDEZ, B.; RUIZ, A.; TORO, M. **An XML-based approach for the automatic verification of software requirements specifications**. In: Fourth Workshop on Requirements Engineering, 2001, Buenos Aires, Argentina. **Proceedings** ... Buenos Aires: Universidad Tecnológica Nacional, 2001. p. 181-194. Disponível em: <http://www.inf.puc-rio.br/wer01/>. Acesso em 23 dez. 2011.

FUXMAN, A.; MYLOPOULOS, J.; PISTORE, M.; TRAVERSO, P. **Model Checking Early Requirements Specifications in Tropos**. In: RE-2001, 9th IEEE International Requirements Engineering Conference, 2001, Toronto, Canada. **Proceedings** on Fifth IEEE International Symposium. Toronto, Canada:[s.n.], 2001.

GIARRATANO, J. C. **CLIPS User's Guide**. Version 6.0. NASA Lyndon B. Johnson Space Center, Software Technology Branch, Houston, Texas, EUA, 1993.

HOLZNER B.; HOLZNER L. **Transparency in Global Change:** The Value of Open society. 1. ed. Pittsburgh, PA, EUA: Universidad of Pittsburgh Press, 2006.

KIYAVITSKAYA, Nadzeya; ZENI, Nicola; MICH, Luisa; BERRY, Daniel M. **Requirements for Tools for Ambiguity Identification and Measurement in Natural Language Requirements Specifications**. In: Workshop em Engenharia de

Requisitos - WER07, 2007, Toronto. **Proceedings** .... Toronto: York University, 2007. p. 197 - 206. Disponível em: <http://wer.inf.puc-rio.br/WERpapers/papers_by_conference.lp?conference=WER07>. Acesso em: 23 dez. 2011.

LAPOUCHNIAN, A. **Goal-Oriented Requirements Engineering:** An Overview of the Current Research. **Technical Report**: Department of Computer Science, University of Toronto, Canadá, 2005. Disponível em: <http://www.cs.toronto.edu/~alexei/pub/Lapouchnian-Depth.pdf>. Acesso em: 25 set. 2010.

LEAL, A. L. C.; SOUSA, H. P.; LEITE, J. C. S. P.; BRAGA, J.L. **Transparência aplicada a modelo de negócios**. In: XIV WER - Workshop em Engenharia de Requisitos (XIV CibSE - Congresso Ibero-Americano em Engenharia de Software), 2011, Rio de Janeiro. **Proceedings** XIV CibSE. Rio de Janeiro, 2011. v. XIV. p. 321-332. Disponível em: <http://wer.inf.puc-rio.br/WERpapers/papers_by_conference.lp?conference=WER11>. Acesso em: 20 jan. 2012.

LEAL, A. L. de C.; ALMENTERO, Eduardo; CUNHA, Herbert; SOUSA, H. P.; LEITE, J. C. S. P. **Bula de Software:** Uma Estrutura Definida para Promover a Melhoria da Transparência em Software**.** In: XV Workshop on Requirements Engineering - WER2012, 2012, Buenos Aires, Argentina. **Anais** ... Buenos Aires, Argentina: UNLaM - Universidad Nacional de La Matanza, 2012. v. 15. Disponível em: < http://wer.inf.puc-rio.br/WERpapers/papers_by_conference.lp?conference=WER12>. Acesso em 11 jun. 2012.

LEITE, J. C. S. P.; WERNECK, V.; PADUA, A.; CAPPELLI, C.; CERQUEIRA, A.; CUNHA, H. S.; GONZÁLEZ, B. **Understanding the Strategic Actor Diagram:** An Exercise of Meta Modeling. In: Workshop on Requirements Engineering, 2007, Toronto, Canada. **Proceedings**... Toronto: York University, 2007. p. 2-12. Disponível em:<http://wer.inf.puc-rio.br/WERpapers/papers_by_conference.lp?conference=WER07>. Acesso em: 22 dez. 2011.

LEITE, J.C.S.P.; CAPPELLI, C.; **Exploring i* Characteristics that Support Software Transparency**. In: International i* Workshop, 2008, Recife, Brasil. **Proceedings**... Recife, Brasil: Universidade Federal de Pernambuco, 2008. v. 322. p. 51-54. Disponível em: <http://ceur-ws.org/Vol-322/fullProceedings.pdf >. Acesso em: 22 jan. 2010.

LEITE, J.C.S.P.; WERNECK, V. M. B; OLIVEIRA, A. Pádua. **Comparing GORE Frameworks:** i-star and KAOS. In: Workshop em Engenharia de Requisitos – WER09, 2009, Valparaíso, Chile. **Proceedings** ... Valparaíso, Chile: Universidad Tecnica Federico Santa Maria, 2009. p. 15-26. Disponível em: <http://wer.inf.puc-rio.br/WERpapers/papers_by_conference.lp?conference=WER09>. Acesso em: 15 jan. 2010.

LUGER, George F. **Inteligência Artificial:** estruturas e estratégias para solução de problemas complexos. 4. ed. Porto Alegre: Bookmann, 2004.

NEGNEVITSKY, M. **Artificial Intelligence:** a guide to intelligent systems. 2. ed. Londres: Pearson Education Limited, 2004.

NEWELL, A.; SIMON, H. A. **Human Problem Solving**. Englewood Cliffs, New Jersey, EUA: Prentice Hall, 1972.

OLIVEIRA, A. P. A.; CAPPELLI, C.; CUNHA, H. S.; LEITE, J. C. P.; WERNECK, V. M. B. **Engenharia de software Intencional: Tornando o Software Mais Transparente.** In: Simpósio Brasileiro de Engenharia de Software, 2007, Rio de Janeiro. Tutorial apresentado no XXI Simpósio Brasileiro de Engenharia de Software, Rio de Janeiro, 2007. Disponível em: <http://www.sbbd-sbes2007.ufpb.br/tuto3.pdf>. Acesso em: 22 jan. 2011.

SILVA, C.; BORBA, C.; CASTRO, J. **A Goal Oriented Approach to Identify and Configure Feature Models for Software Product Lines**. In: Workshop em Engenharia de Requisitos – WER11, (XIV CibSE - Congresso Ibero-Americano em Engenharia de Software), 2011, Rio de Janeiro. **Proceedings** XIV CibSE. Rio de Janeiro, Brasil, 2011. v. XIV. p. 321-332. Disponível em: <http://wer.inf.puc-rio.br/WERpapers/papers_by_conference.lp?conference=WER11>. Acesso em: 20 jan. 2012.

WORDNET: A lexical database for English. Disponível em: <http://wordnet.princeton.edu/wordnet/>. Acesso em: 02 maio 2012.

YU, E.; LIN, L.; LI, Y.; **Modelling Strategic Actor Relationships to Support Intellectual Property Management**. In: International Conference on Conceptual Modeling: Conceptual Modeling, 2001. **Proceedings**... Springer-Verlag, 2001. p. 164-178.

YU, Eric. Agent-Oriented Modelling: Software Versus the World: Agent-Oriented Software Engineering. In: Agent-Oriented Software Engineering – AOSE, Montreal, Canadá, 2001. **Proceedings**... Springer-Verlag, 2001. p. 206.

YU, Eric. **Modelling Strategic Relationships for Process Reengineering.** 1995. 124 f. Tese de Doutorado – Dept. of Computer Science, University of Toronto, Toronto, 1995.